IBM WebSphere Business Integration Adapters

Adapter for WebSphere MQ Workflow User Guide

Adapter Version 2.5.0

Note!
Before using this information and the product it supports, read the information in "Notices" on page 119.

19December2003

This edition of this document applies to the adapter for WebSphere MQ Workflow version 2.5.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

Contents

About this document

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

This document describes the installation, configuration, and business object development for the adapter for WebSphere MQ Workflow.

Audience

This document is for consultants, developers, and system administrators who support and manage the product at customer sites.

Prerequisites for this document

Users of this document should be familiar with the WebSphere business integration system, with business object and collaboration development, with the WebSphere MQ Workflow application and with the run- and buildtime components of WebSphere MQ Workflow.

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

New in this release

New in release 2.5.0

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

Beginning with this release, the adapter for WebSphere MQ Workflow is no longer supported on Microsoft Windows NT.

New in release 2.4.x

The adapter can now use WebSphere Application Server as an integration broker.

The name `MQWorkflow` has been renamed to `WebSphereMQWorkflow` in startup scripts, filenames, and directories.

The adapter now runs on the following platforms:

- Solaris 7, 8
- AIX 5.x
- HP UX 11.i

This release of the adapter requires IBM WebSphere MQ Workflow 3.3.2 or 3.4, and no longer supports the Java API communication mode.

New in release 2.3.x

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

The guaranteed event delivery feature has been enhanced. For further information, see "Enabling guaranteed event delivery" on page 27.

New in release 2.2.x

With this release the connector supports MQ Workflow 3.3.2 XML API verb processing. For more information, see "XML API verb processing" on page 10 and "Upgrading WebSphere MQ Workflow to use XML APIs" on page 20. A new connector-specific property has been added to support the feature; see "JavaCorbaApi" on page 25.

New in release 2.1.x

The connector has been internationalized. For more information, see "Processing locale-dependent data" on page 15 and Appendix A, "Standard configuration properties for connectors," on page 75.

This guide provides information about using this adapter with ICS.

Note: To use the guaranteed event delivery feature, you must install release 4.1.1.2 of ICS.

New in release 1.2.x

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

- An application component specific to MQ Workflow
- Sample business objects
- IBM WebSphere Adapter Framework, which consists of:
  - Connector Framework
  - Development tools (including Business Object Designer and IBM CrossWorlds System Manager)
  - APIs (including CDK)

This manual provides information about using this adapter with ICS.

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

The connector is now enabled for AIX 4.3.3 Patch Level 9.

Chapter 1. Overview

The connector for WebSphere MQ Workflow is a runtime component of the WebSphere Business Integration Adapter for WebSphere MQ. WebSphere MQ Workflow is an IBM workflow management system.

The connector allows the WebSphere integration broker to exchange business objects with WebSphere MQ Workflow. It communicates with WebSphere MQ Workflow client processes (nodes) and with external applications such as the connector described in this book.

This chapter describes the connector component and the business integration system architecture. It covers these topics:

Connector architecture

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

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

The connector for WebSphere MQ Workflow bridges collaborations and WebSphere MQ Workflow nodes by performing exchanges of data structures and controlling related processes. Although the connector is an external application, it functions much like an internal "node" in an WebSphere MQ Workflow system. Like a node, the connector performs a process-oriented function.

WebSphere MQ Workflow nodes are configured to issue requests to the connector on a designated queue. The connector polls and retrieves an WebSphere MQ Workflow request message from such a queue. Using document object model (DOM) parsers and the XML data handler, the connector extracts and converts the data to a business object appropriate to the request and to the corresponding collaboration. In the opposite direction, the connector receives business object requests from collaborations, converts them into MQ Workflow process requests, and issues them to the XML input queue of MQ Workflow. In addition, the connector can directly bind with the MQ Workflow server to provide greater control over MQ Workflow processes.

You configure the connector to use the XML message API of the MQ Workflow system. The XML API of the MQ Workflow system is for asynchronous and synchronous processing of message requests. You create a User-defined Program Execution Server (UPES)—an IBM MQ Workflow program used to trigger workflow actions—that allows MQ Workflow nodes to communicate with the connector as if it were another node. Using the XML API, the connector responds to messages that request business objects from collaborations and conveys messages that request actions in MQ Workflow.

All XML messages exchanged with MQ Workflow conform to a single Document Type Definition (DTD), `WfMessage`. The Document Object Model Parser (DOM) parser extracts the Workflow data structure from the `WfMessage` that the connector then uses to create a container object to hold the data structure.

Note: Although an MQ Workflow process can have different input and output data structures, any given transaction between a collaboration and a connector can involve only one object type. To circumvent this limitation, the connector for MQ Workflow requires that a container object be constructed that has, as children, a request object and one or more response objects. For more on this, see "Meta-object configuration" on page 32.

During polling, the connector knows which top-level object to create by identifying the data structure contained in the `WfMessage`. Specifically, the name of the data structure appended to the *<boprefix>* configuration property determines which top-level object is created. The first (non metadata related) child object in this top-level object is populated with the data structure. The verb assigned to the parent container object is based on the `ProgramParameters` field in the `WfMessage`. The parent container object is posted to InterChange Server.

Application-connector communication method

As noted in the introduction, the connector for MQ Workflow supports the XML API communication mode.

Using the XML API, the connector can send messages that trigger actions in MQ Workflow and handle synchronous requests from MQ Workflow during connector polling. The connector polls a fixed queue to which request MQ messages are issued, processes the content, and returns response messages (to a second queue if necessary). In addition to business content, any XML message issued to the connector indicates the collaboration to execute, the verb to use, and other processing information.

To use the XML API, you must configure a UPES that specifies the input queue of the connector.

The connector and the UPES

When using the XML message API, the connector does not directly poll MQ Workflow to check for new events. You must configure MQ Workflow nodes to issue requests to external queues such as that of the connector. The connector then can poll these external queues.

You configure the MQ Workflow nodes to issue requests to external queues via a UPES. A UPES is a program that is designed to accept requests from the MQ Workflow server. A UPES can, in turn, interact with the server to retrieve any

additional data and to pass results back. MQ Workflow handles the conversion of the request to an XML message and vice-versa.

As shown in Figure 1, the connector acts much like a UPES node in an MQ Workflow system. In fact, the connector is transparent to the MQ Workflow server.

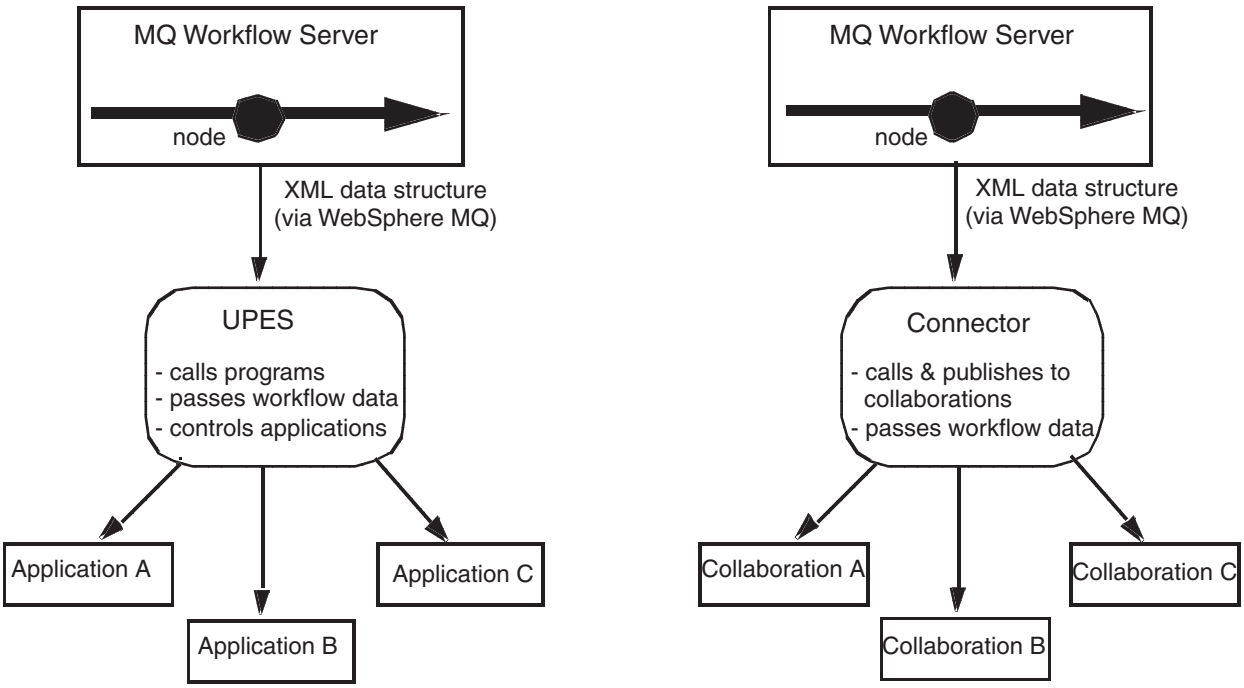

Figure 1. Connector function as an MQ Workflow UPES

Connector-initiated requests: XML API

When the connector receives a message on behalf of a collaboration, the connector issues an XML request message to the XML input queue of the MQ Workflow server. Optionally (synchronously), the connector waits for a response message to be returned by the MQ Workflow server. The server triggers a workflow process and issues a response as necessary.

Figure 2 illustrates a message request communication from a collaboration via the connector to an MQ Workflow.

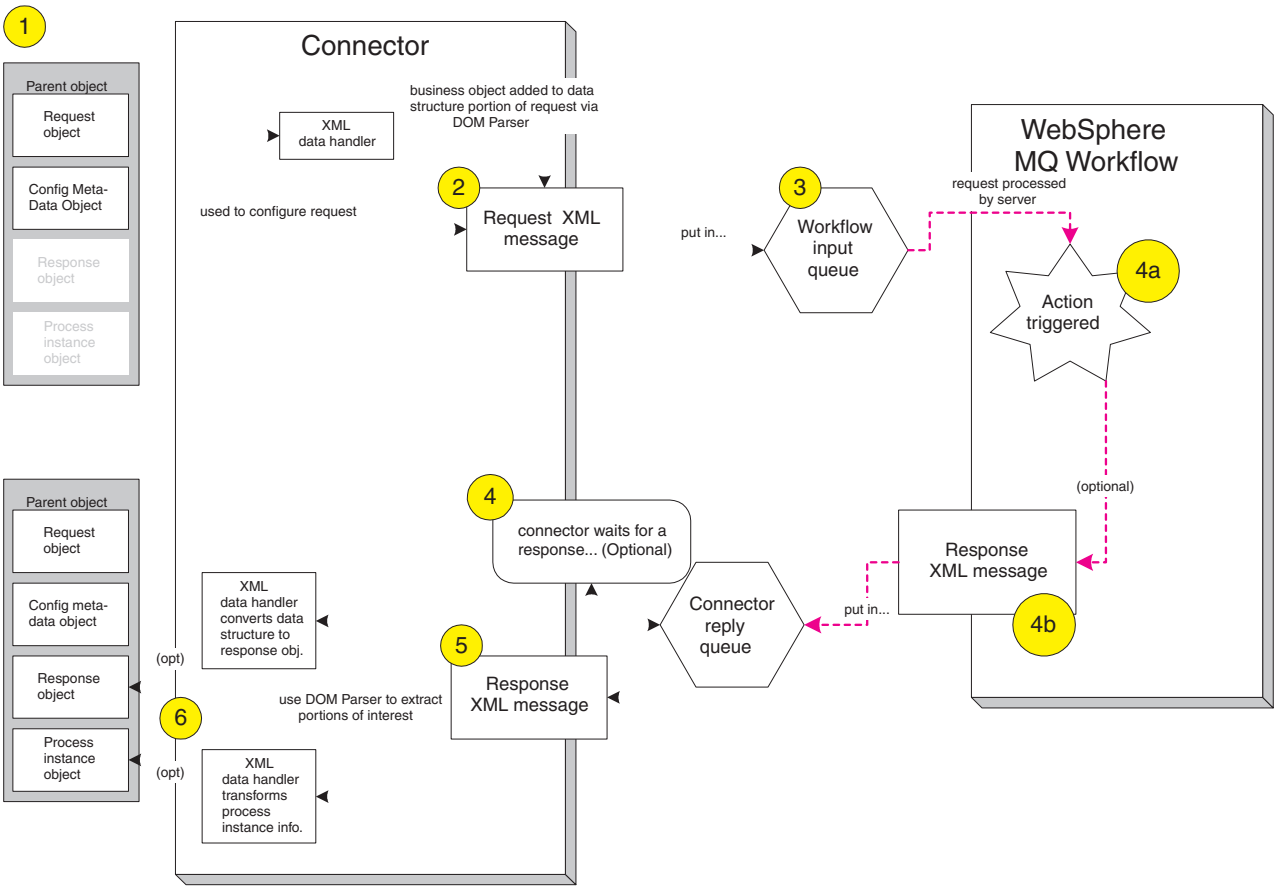

Figure 2. XML API communication mode: Connector-initiated request

1. The connector receives a request from a collaboration for a business object destined for MQ Workflow.
2. The connector converts the request object contained in the parent business object to XML using the XML data handler. Using a DOM parser, the XML-serialized business object is incorporated into a larger XML message conforming to the MQ Workflow message DTD. Information about which workflow process to create and execute is encapsulated in the (configuration meta-object) that is contained in the parent business object.
3. The connector posts the request to the XML input queue of the MQ Workflow server.
4. The MQ Workflow server receives the request and performs an action as specified by the business object content. Running in parallel, the connector either a) return successfully (if no response is expected) or b) optionally, wait for a response message.
5. Depending on the request message issued by the connector, the MQ Workflow server may return a response immediately that contains only the process instance identifier (PID) of the concurrently executing process. Or the MQ Workflow server may wait until the process is complete before returning the resulting output data structure.
6. Using an XML DOM parser, the connector extracts the output data structure and process ID from the response message. As necessary, these structures are converted to business objects using the XML data handler and added to the parent business object. This object is then returned to the awaiting collaboration.

MQ Workflow-Initiated requests

Figure 3 illustrates an MQ Workflow-initiated request to the connector.

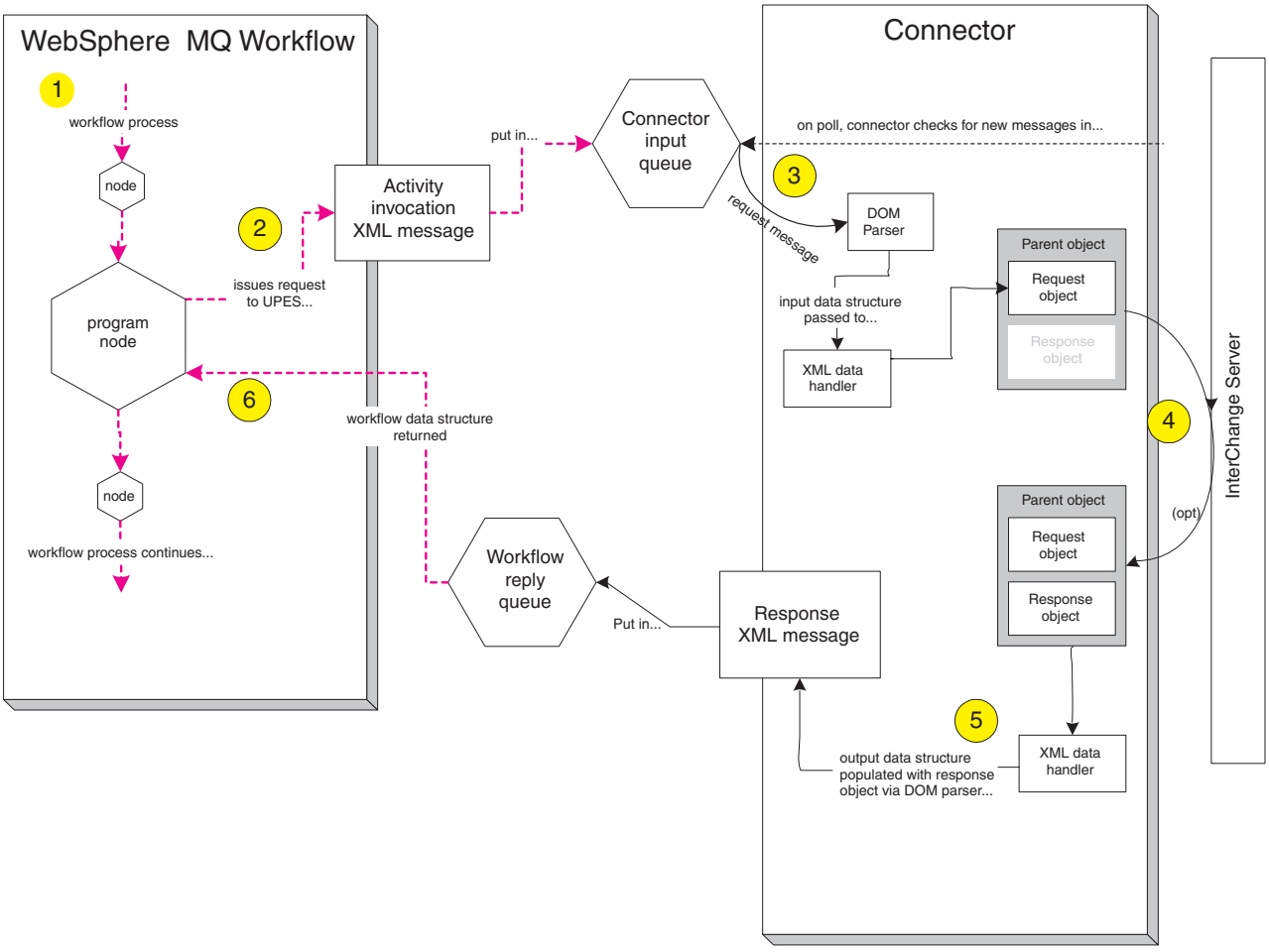

Figure 3. MQ Workflow-Initiated request

1. During workflow implementation, a UPES is defined such that messages sent to it are issued to the connector input queue.
2. To request an action from a collaboration, the MQ Server routes the request data structure to the UPES (as indicated by "program node" above). MQ Workflow then sends an XML-based program invocation message to the designated connector input queue. The message contains the workflow data structure.
3. The connector retrieves the message while polling the queue, parses the content using an XML DOM parser and converts the input data structure to a business object using the XML Data Handler.
4. If the request is to execute asynchronously, as specified by the user when defining the UPES (program node), the connector posts the business object to all waiting collaborations and does not send a response back to the MQ Workflow server.
5. If the request is to execute synchronously and the collaboration name is specified in the command line parameters (see Figure 20 on page 56), the connector sends the business object to the specific collaboration and waits for a

return.The connector constructs an appropriate response message based on the return. The response contains any business object changes. The response is then sent to the MQ Workflow server.

6. If the request is to execute synchronously and the collaboration name is not specified in the command line parameters (see Figure 20 on page 56), the connector posts the business object to all waiting collaborations and does not send a response back to the MQ Workflow server. Although this case is similar to the asynchronous case above, the MQ Workflow server is still waiting for the reply from the connector. Therefore it is the responsibility of collaboration to send the corresponding response to the connector using the service call request.
7. The requesting MQ Workflow UPES (program node) receives the response code and resulting business data structure.

Event handling

An event occurs when an MQ Workflow UPES posts a request to the connector input queue. The connector detects events by polling the input queue. This section describes the Event Handling process.

Event notification

After detecting an event, the connector performs the steps described below.

1. The connector loads the XML message into a DOM parser.
2. The connector verifies that this is a WfMessage and checks for a recognized template.
3. After identifying the input data structure contained in the message, the connector creates a business object of type <boprefix><*data structure name*>. For example, if connector configuration property *boprefix* has a value of `WfRequest_` and the connector receives a message containing a data structure named `MyCustomer`, the connector expects that structure to conform to a child of the top-level business object `WfRequest_MyCustomer`. The connector uses the XML data handler to convert the XML data structure to the business object.
4. The connector locates element ProgramParameters in the WfMessage. The ProgramParameters field contains application parameters specified by the user in the actual MQ Workflow.
5. If the business object created by the data handler does not have a verb specified, the connector sets the verb using data specified in the ProgramParameters portion of the request message.
6. To determine whether MQ Workflow expects a response to its request, the connector evaluates element ResponseRequired in container WfMessageHeader of the XML document for a value of `yes`, `no`, or `iferror`. A value of `yes` indicates that MQ Workflow is actively waiting for a response message from the connector on the status of the request.

   Note: The ResponseRequired element corresponds to the mode in the program activity properties of the MQ Workflow Buildtime configuration (see Defining the Workflow Server in Figure 22 on page 58). If you specify the asynchronous mode, then `ResponseRequired=no`; if you specify synchronous mode, then `ResponseRequired=yes`.
7. If a response is expected (`ResponseRequired=yes`), the connector evaluates the data specified in the ProgramParameters element in the WfMessage to determine how to handle the request.
8. If a collaboration is specified in ProgramParameters, the connector sends the request to the collaboration using the `executeCollaboration()` method.

Because this method is a synchronous request to the collaboration, the return from the method call may take time. The method returns the response object in its argument. The connector generates a response message by populating the response object, then sends the response to the MQ Workflow server.

9. If a collaboration is not specified in ProgramParameters, the connector posts the request to all subscribing collaborations using the gotApplEvent() method. The connector populates the business object shown in the argument of the gotApplEvent() method with the meta-object that contains the MQ Workflow Activity information (such as ActImplCorrelID). The method posts the request to the collaboration, so the connector receives the return from the method call immediately. Since no response object is returned by the method call, no response message is generated or sent to the MQ Workflow server. Instead, the collaboration must send the corresponding response object to the MQ Workflow server using the service call request. The response object must include the meta-object that contains the MQ Workflow Activity information (such as ActImplCorrelID). The connector then constructs an appropriate response message based on the service call request. The response message contains the MQ Workflow Activity information and the return code from the collaboration as well as the response business object. The response message is sent to the MQ Workflow server. If an error occurs at the step 8 or 9 above due to problems unrelated to the business content of the message, the connector logs the error and does not generate a response WfMessage.
10. The message is optionally archived in the archive, error, or unsubscribed queue.

Retrieval

The connector polls its input queue at regular intervals for messages. When the connector finds a message, it retrieves it from the queue and passes it to the DOM parser and XML data handler to obtain the `WfMessage` content. The connector uses the data structure extracted from the `WfMessage` to generate an appropriate business object with a verb. See "Error handling" on page 66 for event failure scenarios.

The connector processes messages by first opening a transactional session to the input queue. This transactional approach allows for the small chance that a business object could be delivered to a collaboration twice due to the connector successfully submitting the business object but failing to commit the transaction in the queue. To avoid this problem, the connector moves all messages to an in-progress queue. There, the message is held until processing is complete. If the connector shuts down unexpectedly during processing, the message remains in the in-progress queue instead of being reinstated to the original input queue.

Note: The connector does not remove the message from the in-progress queue until: 1) The message has been converted to a business object 2) the business object is delivered to InterChange Server, and 3) a return value is received.

Recovery

Upon initialization, the connector checks the in-progress queue for messages that have not been completely processed, presumably due to a connector shutdown. The connector configuration property `InDoubtEvents` allows you to specify one of four options for handling recovery of such messages: fail on startup, reprocess, ignore, or log error.

FailOnStartup

With the FailOnStartup option, if the connector finds messages in the in-progress queue during initialization, it logs an error and immediately shuts down. It is the

responsibility of the user or system administrator to examine the message and take appropriate action, either to delete these messages entirely or move them to another queue.

Reprocess

With the reprocessing option, if the connector finds any messages in the in-progress queue during initialization, it processes these messages first during subsequent polls. When all messages in the in-progress queue have been processed, the connector begins processing messages from the input queue.

Ignore

With the ignore option, if the connector finds any messages in the in-progress queue during initialization, the connector ignores them, but does not shut down. It begins processing messages from the input queue.

Log error

With the log error option, if the connector finds any messages in the in-progress queue during initialization, it logs an error but does not shut down. It begins processing messages from the input queue.

Archiving

If the connector property `ArchiveQueue` is specified and identifies a valid queue, the connector places copies of all successfully processed messages in the archive queue. If `ArchiveQueue` is undefined, successfully processed messages are discarded after processing. For more information on archiving unsubscribed or erroneous messages, see "Error handling" on page 66 in Chapter 4, "Developing business objects," on page 59.

Guaranteed event delivery

The guaranteed-event-delivery feature enables the connector framework to ensure that events are never lost and never sent twice between the connector's event store, the JMS event store, and the destination's JMS queue. To become JMS-enabled, you must configure the connector`DeliveryTransport` standard property to JMS. Thus configured, the connector uses the JMS transport and all subsequent communication between the connector and the integration broker occurs through this transport. The JMS transport ensures that the messages are eventually delivered to their destination. Its role is to ensure that once a transactional queue session starts, the messages are cached there until a commit is issued; if a failure occurs or a rollback is issued, the messages are discarded.

Note: Without use of the guaranteed-event-delivery feature, a small window of possible failure exists between the time that the connector publishes an event (when the connector calls the gotApplEvent() method within its pollForEvents() method) and the time it updates the event store by deleting the event record (or perhaps updating it with an "event posted" status). If a failure occurs in this window, the event has been sent but its event record remains in the event store with an "in progress" status. When the connector restarts, it finds this event record still in the event store and sends it, resulting in the event being sent twice.

You can configure the guaranteed-event-delivery feature for a JMS-enabled connector with, or without, a JMS event store. To configure the connector for guaranteed event delivery, see "Enabling guaranteed event delivery" on page 27.

If connector framework cannot deliver the business object to the ICS integration broker, then the object is placed on a FaultQueue (instead of UnsubscribedQueue and ErrorQueue) and generates a status indicator and a description of the problem. FaultQueue messages are written in MQRFH2 format.

Business object requests

Business object requests are processed when InterChange Server sends a business object to the `connector`. Using the XML data handler, the connector converts the business object to an MQ Workflow `WfMessage` formatted message and issues it to MQ Workflow.

The `WfMessage` contains the business object and processing information, and is encapsulated in an MQ Message with an MQ Message Descriptor (MQMD) header. The MQMD header contains a `UserID` field that the connector populates. (By default, the connector passes the value of the connector configuration property `ApplicationUserID`.) MQ Workflow uses the `UserID` field in the MQMD header of the MQ Message to perform authorization checks on any requests sent by the connector.

Note: If the attribute `UserID` is defined in the child configuration meta-object that populates the `WfMessage` data structure, this `UserID` attribute overrides the value defined statically in the connector configuration property for the instance of the business object issued to MQ Workflow.

Note: Although a user may have authorization to initiate a workflow process in MQ Workflow, the user still needs authorization to issue a message via WebSphere MQ. See WebSphere MQ documentation on how to authorize users to issue messages.

Verb processing

The role of the connector is to bridge the data structures in the workflow with the business object. It is the responsibility of MQ Workflow to take action depending on the verbs set in the business object—the connector can guarantee only that the workflow successfully receives the content. Accordingly, the connector has no means of influencing a business object in the MQ Workflow product. In fact, given the nature of MQ Workflow, a persistent data structure may not exist for the business object as it may act only as a trigger for subsequent workflow.

The Business Object Handler (BOHandler) processes all business objects in the same manner regardless of the verb. Using a DOM parser, the connector constructs a `WfMessage`.

The connector checks the application-specific text of the top-level business object to ensure that a name-value pair of the form `cw_mo_wfptcfg=`*`XXX`* is defined. The child meta-object identified by *`XXX`* is parsed and values are interpreted.

The template that is executed is identified by meta-object attribute `ProcessTemplateName`. These templates provide the structures necessary to specify entire commands to MQ Workflow as well as to contain the results. If attribute `ProcessInstanceName` is specified, the connector executes the existing instance; otherwise, it creates a new instance of the template. The template is executed under the authorities granted to the user identified by attribute `UserId` of the child meta-object. If the attribute is not specified, the value of connector configuration property `ApplicationUserID` is used instead. For more on templates, see "Top-level

business object and content configuration" on page 31 in Chapter 3, "Modifying the WebSphere MQ Workflow application," on page 51.

The connector supports the following business object verbs for requests to control an existing workflow:

- Delete
- Suspend
- Terminate
- Restart
- Resume

Upon receipt of a message from MQ Workflow, the connector identifies the verb of the business object from the value of `ProgramParameters` in the `WfMessage`. The text for `ProgramParameters` must include a *`name=value`* pair specifying the verb of the business object contained in the message. For example, to specify a delete verb, the element text includes the *`name=value`* pair `verb=Delete`.

However, in the opposite direction, the connector does not dictate to MQ Workflow which verb to use to process the request. When issuing a business object to MQ Workflow, the connector ignores the verb of the business object. Instead, the connector converts the business object to XML and incorporates the content into a `WfMessage` for MQ Workflow. The workflow to which the business object is issued determines the action taken (not the verb specified for the business object by the collaboration)

XML API verb processing

Note: For MQ Workflow version 3.3.2 and higher, the XML API is recommended for verb processing.

Using the MQ Workflow connector XML API, a collaboration can monitor and control the status of the workflow process. Upon successfully controlling a workflow operation, the connector populates a process instance object (`MO_MQWorkflow_ProcessInstance`) with details of the process. The connector considers any object with `app-text` equal to `ProcessInstance` to be an instance of an `MO_MQWorkflow_ProcessInstance`.

The connector converts the business object to XML and incorporates the content into a WfMessage of MQ Workflow. The verb specified for the business object determines the action performed on the process instance.

When using the XML API, the connector supports the following verbs for an `MO_MQWorkflow_ProcessInstance`:

Delete

The connector deletes the specified process instance from MQ Workflow. The process instance must be in one of the following states: Ready, Finished, or Terminated. If the process does not exist or could not be deleted, the connector returns BON_FAIL. Otherwise, the connector returns a populated `MO_MQWorkflow_ProcessInstance` object with new status.

Suspend

The connector issues a request to suspend the workflow process and returns BON_FAIL if the process does not exist or if it cannot be suspended. The process instance must be in the state of Running. If the deep option is true, all

non-autonomous sub-processes are also suspended. If the process does not exist or could not be suspended, the connector returns BON_FAIL. Otherwise, the connector returns a populated `MO_MQWorkflow_ProcessInstance` object with new status.

Terminate

The connector terminates a process instance and all of its non-autonomous sub-processes. All running, checked-out, and suspended activities are terminated. The process must in one of the following states: Running, Suspended, or Suspending. If the process does not exist or cannot be terminated, the connector returns BON_FAIL. Otherwise, the connector returns a populated `MO_MQWorkflow_ProcessInstance` object with new status.

Restart

The connector issues a request to restart the workflow process instance. Only finished or terminated top-level process instances can be restarted. If the process does not exist or could not be restarted, the connector returns BON_FAIL. Otherwise, the connector returns a populated `MO_MQWorkflow_ProcessInstance` object with new status.

Resume

The connector issues a request to resume processing of a suspended or suspending process instance. If the deep option is true, all non-autonomous sub-processes are also resumed. If the process does not exist or could not be resumed, the connector returns BON_FAIL. Otherwise, the connector returns a populated `MO_MQWorkflow_ProcessInstance` object with new status.

Asynchronous requests

If the attribute `ResponseTimeout` of the configuration meta-object is less than zero, the connector issues requests to the MQ Workflow server without waiting for a response. If an error occurs while the process is executing, the collaboration has no means of being notified. Figure 4 shows a sample asynchronous request.

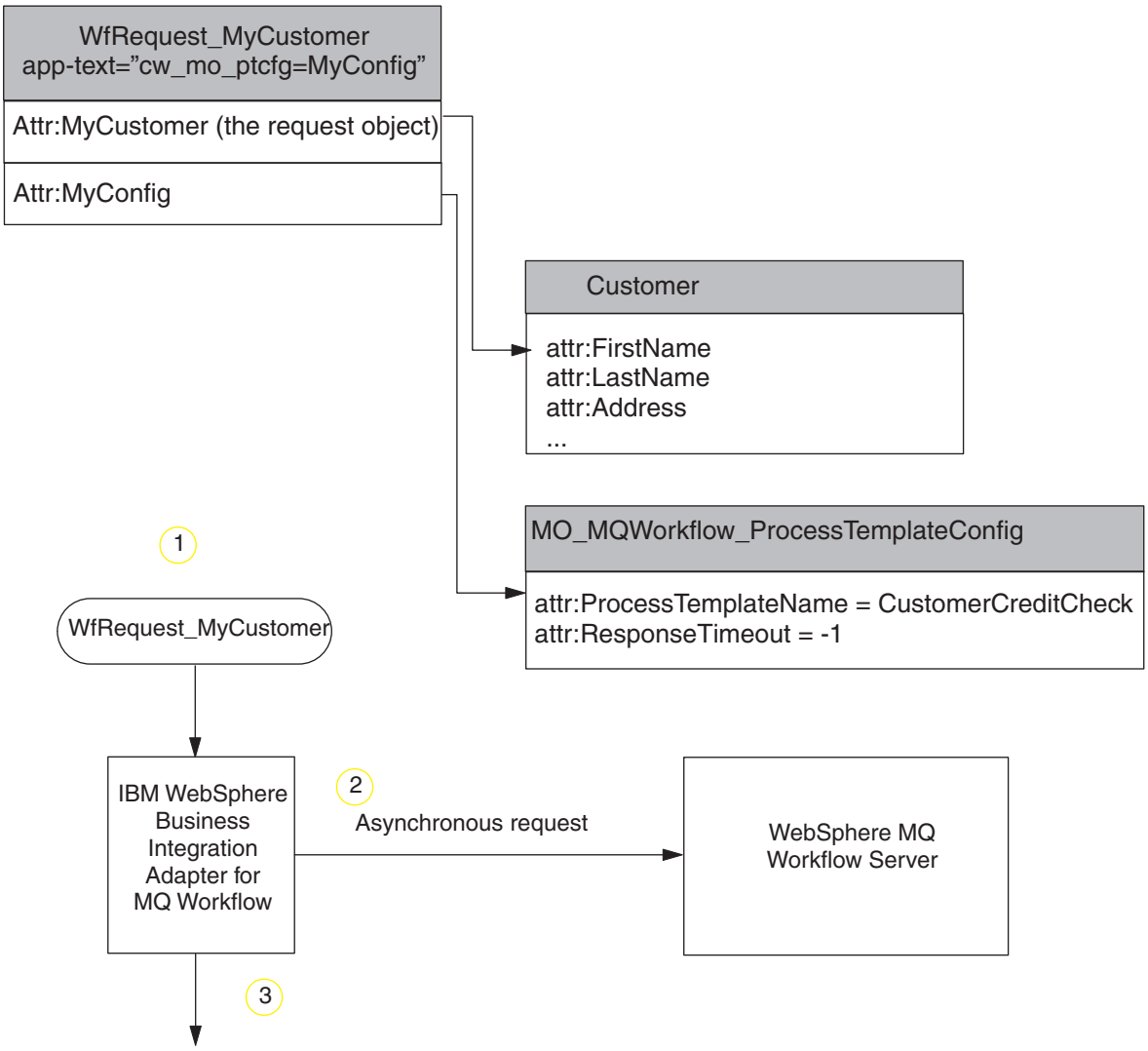

Figure 4. Sample asynchronous connector request to MQ Workflow

1. The connector receives a `WfRequest_MyCustomer` top-level business object with a negative "ResponseTimeout" attribute.
2. The connector issues a request to the MQ Workflow server containing object "MyCustomer" as the data structure to be passed to the process.
3. The connector returns successfully without waiting for a response. An error occurs only if the connector fails to put a message in the XML input queue of the MQ Workflow server.

Asynchronous requests for a process instance ID

If a non-negative `ResponseTimeout` is provided in the child meta-object and attribute `ExecutionMode` is *`Asynchronous`*, the connector issues a request and returns a process instance ID to the collaboration. Successful receipt of a process instance ID does not imply successful completion of the corresponding workflow process. The collaboration must perform a "Retrieve" on the process instance ID to determine the status. This is useful for long-lived transactions. Figure 5 illustrates this process.

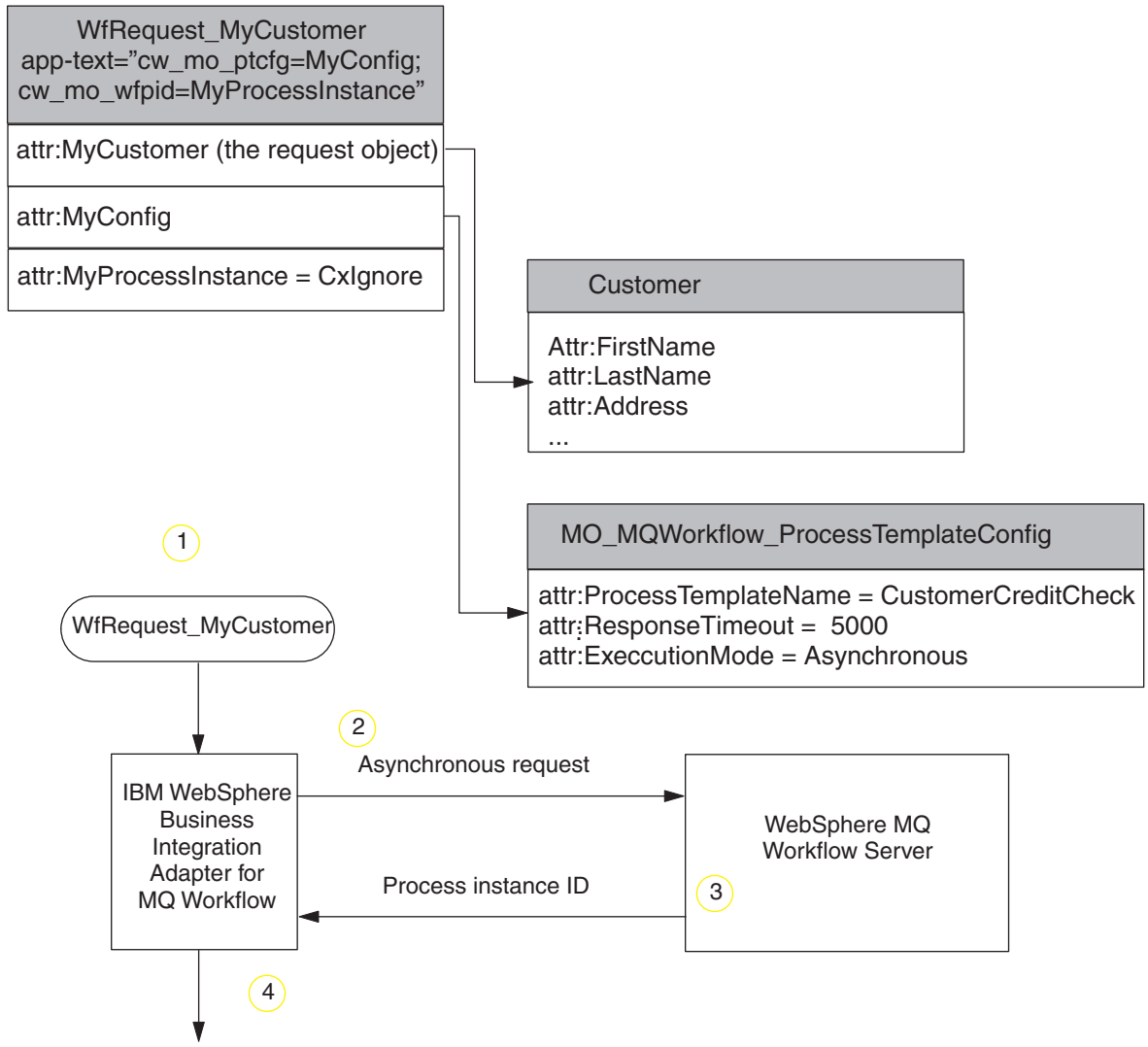

Figure 5. Sample asynchronous connector request for a process instance ID

1. The connector receives a `WfRequest_MyCustomer` top-level business object that specifies a non-negative value for attribute *`ResponseTimeout`* and the value *`Asynchronous`* for attribute `ExecutionMode`.
2. The connector issues a request message to the MQ Workflow server containing object `MyCustomer` as the input data structure for process template `CustomerCreditCheck`. The connector waits (up to 5000 ms) for a reply.
3. A new MQ Workflow process instance `CustomerCreditCheck` is instantiated. Before the workflow process finishes, a response message is returned to the connector. The message contains only a process instance ID.
4. The connector populates object `MyProcessInstance` with the process instance information. If MQ Workflow returns an error message, the connector returns BON_FAIL and conveys the error message provided in the workflow message.

Synchronous requests

When a non-negative `ResponseTimeout` attribute is provided in the child meta-object and attribute `ExecutionMode` is *`Synchronous`*, the connector issues a synchronous request. The request does not return successfully until the workflow process has completed. Synchronous request processing guarantees that a collaboration is apprised of the success or failure of an MQ Workflow process that

the collaboration initiates. For short-lived transactions, synchronous processing is an efficient means of generating immediate feedback. Figure 6 illustrates a synchronous request.

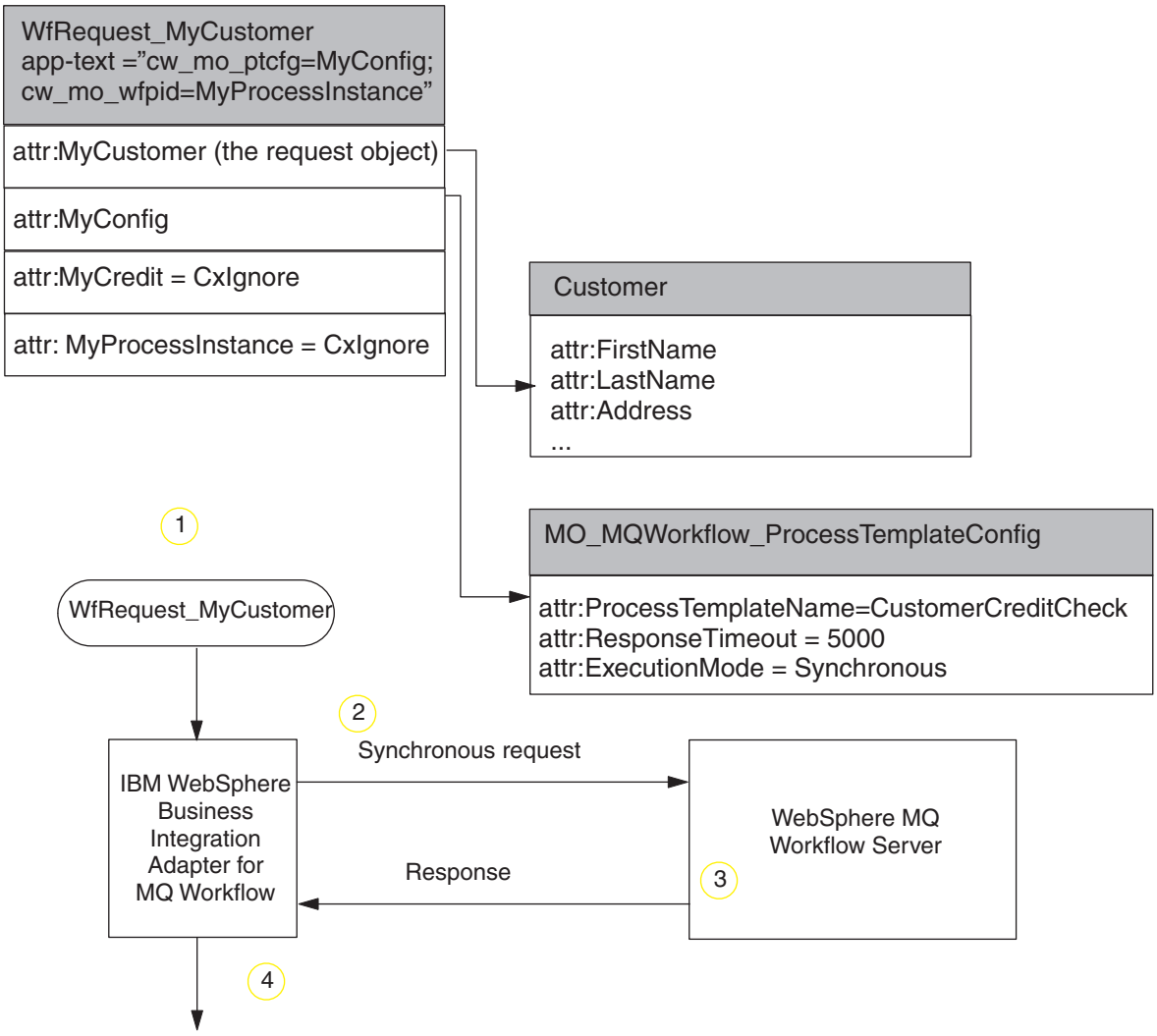

Figure 6. Sample synchronous connector request to MQ Workflow

1. The connector receives a `WfRequest_MyCustomer` top-level business object and the configuration meta-object specifies a non-negative value for attribute `ResponseTimeout` and the value *`Synchronous`* for attribute `ExecutionMode`.
2. The connector issues a request message to the MQ Workflow server containing object `MyCustomer` as the input data structure for process template `CustomerCreditCheck`. The connector waits for a reply (up to 5000 ms).
3. A new instance of workflow process `CustomerCreditCheck` is instantiated in MQ Workflow. When the workflow process finishes, a response message is returned to the connector.

   Note: For long-lived synchronous transactions, ensure that the value for `ResponseTimeout` is sufficient to allow the entire process to execute.
4. The connector transforms the data structure returned by the process to response object `MyCreditCheck`. In addition, the connector populates object `MyProcessInstance` with the process instance information. If MQ Workflow

returns an error message, the connector returns BON_FAIL and conveys the error message provided in the MQ Workflow message.

Processing locale-dependent data

The connector has been internationalized so that it can support double-byte character sets, and deliver message text in the specified language. When the connector transfers data from a location that uses one character code to a location that uses a different code set, it performs character conversion to preserves the meaning of the data.

The Java runtime environment within the Java Virtual Machine (JVM) represents data in the Unicode character code set. Unicode contains encodings for characters in most known character code sets (both single-byte and multibyte). Most components in the WebSphere business integration system are written in Java. Therefore, when data is transferred between most integration components, there is no need for character conversion.

To log error and informational messages in the appropriate language and for the appropriate country or territory, configure the `Locale` standard configuration property for your environment. For more information on configuration properties, see Appendix A, “Standard configuration properties for connectors,” on page 75.

Chapter 2. Installing and configuring the connector

This chapter describes how to install and configure the IBM WebSphere Business Integration adapter for WebSphere MQ Workflow. 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 2.5 of the adapter for WebSphere MQ Workflow is supported on the following adapter framework and integration brokers:

- **Adapter framework**:
  WebSphere Business Integration Adapter Framework versions 2.3.1 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
  - WebSphere Application Server Enterprise, version 5.0.2, with WebSphere Studio Application Developer Integration Edition, version 5.0.1

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.*
For WebSphere Application Server, see *Implementing Adapters with WebSphere Application Server.*

Adapter platforms

Before you install the adapter, your system should have the following software installed and configured:

Operating systems:

One of the following application platforms:

- AIX 5.1, AIX 5.2
- Solaris 7.0, Solaris 8.0
- HP UX 11i
- Windows 2000

Third-party software:

IBM WebSphere MQ Workflow version 3.4

Globalization

This adapter is DBCS (double-byte character set)-enabled.

Prerequisites

You must set up a client with the Windows 2000 server. To do this, refer to the document *WebSphere MQ Quick Beginnings for Windows 2000*.

Overview of installation tasks

To install the connector for WebSphere MQ Workflow, you must perform the following tasks:

- **Install the integration broker** This task, which includes installing the WebSphere business integration system and starting the integration broker, is described in the installation documentation for your broker and operating system.
- **Install the adapter and related files** This task includes installing the files for the adapter from the software package onto your system. See "Installing the adapter and related files."

Installing the adapter and related files

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

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

Installed file structure

The sections below describe the path and filenames for the product after installation

Windows connector file structure

The Installer copies the standard files associated with the connector into your system.

The utility installs the connector agent into the *ProductDir*\connectors\WEBSPHEREMQWORKFLOW directory, and adds a shortcut for the connector agent to the Start menu.

Table 1 describes the Windows file structure used by the connector, and shows the files that are automatically installed when you choose to install the connector through Installer. For instructions on installing, configuring, and running the samples, see Appendix B, "Connector Configurator," on page 91.

Table 1. Installed Windows file structure for the connector

Subdirectory of *ProductDir*	Description
connectors\WebSphereMQWorkflow\CWMQWorkflow.jar	Contains class files used by WebSphere MQ Workflow connector
connectors\WebSphereMQWorkflow\utilities\FdlBorgen.jar	Contains class files used by the FDLBORGEN utility
repository\WebSphereMQWorkflow\CN_WebSphereMQWorkflow.txt	Repository definition for the connector
repository\WebSphereMQWorkflow\WebSphereMQWorkflow_MetaObjects.txt	Repository definition for meta-objects used by the connector
connectors\messages\WebSphereMQWorkflowConnector.txt	Connector message file
connectors\WebSphereMQWorkflow\start_WebSphereMQWorkflow.bat	The startup script for the connector (2000)
connectors\WebSphereMQWorkflow\utilities\fdlborgen.bat	The startup script for the FDLBORGEN utility (NT/2000)
connectors\WebSphereMQWorkflow\Samples\WebSphereMQWorkflow_Samples.fdl	Workflow definition file containing sample workflows
connectors\WebSphereMQWorkflow\Samples\Sample_MQWF_Order_Collaborations.in	Repository definition for sample collaborations
connectors\WebSphereMQWorkflow\Samples\Sample_MQWF_Order_Connectors.in	Repository definition for sample objects
connectors\WebSphereMQWorkflow\Samples\Sample_MQWF_Order_Objects.in	Repository definition for sample objects
connectors\WebSphereMQWorkflow\Samples\collaboration_classes\SampleItemOrderSync.class	Sample collaboration class
connectors\WebSphereMQWorkflow\Samples\collaboration_classes\SampleItemSync.class	Sample collaboration class
connectors\WebSphereMQWorkflow\Samples\collaboration_classes\SampleWorkflowProcessControl.class	Sample collaboration class
connectors\WebSphereMQWorkflow\Samples\collaboration_classes\SampleItemActivityImpl.class	Sample collaboration class
connectors\WebSphereMQWorkflow\Samples\map_classes\MQWF_Sample_GBOtoResponse.class	Sample map class
connectors\WebSphereMQWorkflow\Samples\map_classes\MQWF_Sample_RequesttoGBO.class	Sample map class

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

UNIX connector file structure

The Installer copies the standard files associated with the connector into your system.

The utility installs the connector agent into the *ProductDir*/connectors/WEBSPHEREMQWORKFLOW directory.

Table 2 describes the UNIX file structure used by the connector, and shows the files that are automatically installed when you choose to install the connector through Installer. For instructions on installing, configuring, and running the samples, see Appendix B, "Connector Configurator," on page 91.

Table 2. Installed UNIX file structure for the connector

Subdirectory of *ProductDir*	Description
connectors/WebSphereMQWorkflow/CWMQWorkflow.jar	Contains class files used by WebSphere MQ Workflow connector
connectors/WebSphereMQWorkflow/utilities/FdlBorgen.jar	Contains class files used by the FDLBORGEN utility
repository/WebSphereMQWorkflow/CN_WebSphereMQWorkflow.txt	Repository definition for the connector
repository/WebSphereMQWorkflow/WebSphereMQWorkflow_MetaObjects.txt	Repository definition for meta-objects used by the connector
connectors/messages/WebSphereMQWorkflowConnector.txt	Connector message file
connectors/WebSphereMQWorkflow/start_WebSphereMQWorkflow.sh	System startup script for the connector. This script is called from the generic connector manager script. When you click the Connector Configuration screen of System Manager, the installer creates a customized wrapper for this connector manager script. Use this customized wrapper to start and stop the connector.
connectors/WebSphereMQWorkflow/utilities/fdlborgen.sh	The startup script for the FDLBORGEN utility
connectors/WebSphereMQWorkflow/Samples/WebSphereMQWorkflow_Samples.fdl	Workflow definition file containing sample workflows
connectors/WebSphereMQWorkflow/Samples/Sample_MQWF_Order_Collaborations.in	Repository definition for sample collaborations
connectors/WebSphereMQWorkflow/Samples/Sample_MQWF_Order_Connectors.in	Repository definition for sample objects
connectors/WebSphereMQWorkflow/Samples/Sample_MQWF_Order_Objects.in	Repository definition for sample objects
connectors/WebSphereMQWorkflow/Samples/collaboration_classes/SampleItemOrderSync.class	Sample collaboration class
connectors/WebSphereMQWorkflow/Samples/collaboration_classes/SampleItemSync.class	Sample collaboration class
connectors/WebSphereMQWorkflow/Samples/collaboration_classes/SampleWorkflowProcessControl.class	Sample collaboration class
connectors/WebSphereMQWorkflow/Samples/collaboration_classes/SampleItemActivityImpl.class	Sample collaboration class
connectors/WebSphereMQWorkflow/Samples/map_classes/MQWF_Sample_GBOtoResponse.class	Sample map class
connectors/WebSphereMQWorkflow/Samples/map_classes/MQWF_Sample_RequesttoGBO.class	Sample map class

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

Upgrading WebSphere MQ Workflow to use XML APIs

To upgrade to WebSphere MQ Workflow to make use of the XML APIs (for more information on XML APIs, see "XML API verb processing" on page 10): you must perform the following tasks:

1. Install WebSphere MQ Workflow 3.3.2.
2. Create collaborations to support verbs Delete and Restart as well as Suspend, Terminate, and Resume. The XML APIs support use of all of these verbs.
3. Add connector-specific property JavaCorbaApi and set its value to false.
4. Add the following attributes to the MO_MQWorkflow_ProcessInstance business object:

Table 3. Connector-specific configuration properties

Name	Type	Application-specific information
ProcTemplValidFromDate	String	`ProcTemplValidFromDate; type=pcdata;`
ProcInstSuspensionTime	String	`ProcInstSuspensionTime; type=pcdata;`
ProcInstSuspension ExpirationsTime	String	`ProcInstSuspension ExpirationsTime; type=pcdata;`

Table 3. Connector-specific configuration properties (continued)

Name	Type	Application-specific information
ExternalProcessContext	String	`ExternalProcessContext; type=pcdata` **Note:** If you are using flow monitoring with the InterChange server, leave this attribute undefined (see 7. below).

Or:
repos_copy MQWorkflowMetaObject.txt and CNMQWorkflow.txt from the Repository folder.

5. To process Suspend and Resume verbs, add the following application-specific information to MO_MQWorkflow_ProcessInstance:

```
Suspend      deep=true;
Resume       deep=true;
```

When deep=true, all non-autonomous subprocesses are also suspended or resumed; when deep=false, these same subprocesses are ignored. The default for deep is false.

6. To control or monitor the status of a workflow process, use the ProcInstName (instead of ProcInstID) of the MQ_MQWorkflow_ProcessInstance object.
7. When ExternalProcessContext is left undefined in the object, the InterChange server passes flow monitoring information to the adapter through the ObjectEvent ID.

Connector configuration

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

You use Connector Configurator to configure connector properties:

- For a description of Connector Configurator and step-by-step procedures, see Appendix B, "Connector Configurator," on page 91.
- For a description of standard connector properties, see "Standard connector properties" and then Appendix A, "Standard configuration properties for connectors," on page 75.
- For a description of connector-specific properties, see "Connector-specific properties" on page 22.

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, "Standard configuration properties for connectors," on page 75 for documentation of these properties. When you set configuration properties in Connector Configurator, you specify your broker using the BrokerType property. Once this is set the properties relevant to the broker appear in the Connector Configurator window. For more information, see Appendix B, "Connector Configurator," on page 91.

Connector-specific properties

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

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	Required
ApplicationPassword	*Login password*	password	No
ApplicationUserID	*Login user ID*	ADMIN	No
ArchiveQueue	*Queue to which copies of successfully processed messages are sent*	`MQWFCONN.ARCHIVE`	No
MQSeriesCCSID	*Character set for queue manager connection*	`null`	No
MQSeriesChannel	*MQ server connector channel*	FMCQM.CL.TCP	Yes
DataHandlerClassName	*Data handler class name*	`com.crossworlds .DataHandlers. text.xml`	No
DataHandlerConfigMO	*Data handler meta-object*	`MO_DataHandler _Default`	Yes
DataHandlerMimeType	*MIME type of file*	`text/xml`	No
ErrorQueue	*Queue for messages containing errors*	`MQWFCONN.ERROR`	No
MQSeriesHostName	*Name of machine containing WebSphere MQ Workflow queue manager*	blank	Yes
InDoubtEvents	*FailOnStartup Reprocess IgnoreLogError*	`Reprocess`	No
InputQueue	*Queue to poll for WebSphere MQ Workflow requests*	`CWLDINPUTQ`	Yes
InProgressQueue	*In-progress event queue*	`MQWFCONN.IN_ PROGRESS`	Yes
JavaCorbaApi	*Enables Java CORBA API when its value is set to `true`*	`false`	No
OutputQueue	Queue to issue requests to WebSphere MQ Workflow	FMC.FMCGRP.EXE. XML	Yes
PollQuantity	*Number of messages to retrieve from the input queue*	1	No
MQSeriesPort	*Port established for the WebSphere MQ (MQSeries) listener*	14000	Yes
ReplyToQueue	*Queue to which response messages are delivered when the connector issues requests*	`MQWFCONN.REPLYTO`	No
UnsubscribedQueue	*Queue to which unsubscribed messages are sent*	`MQWFCONN. UNSUBSCRIBE`	No
MQSeriesQueueManager	Queue manager for Workflow	FMC (if left blank, the default queue manager is used)	No
BOPrefix	For subscription deliveries, the name of the data structure is appended to this prefix to determine the name of the top-level business object.	MQWF_	No
WorkflowSystemName	The name of the WebSphere MQ Workflow system to which the connector binds for direct control of a workflow process.	`FMCSYS`	No

Table 4. Connector-specific configuration properties (continued)

Name	Possible values	Default value	Required
WorkflowSystemGroup	The name of the WebSphere MQ Workflow system group to which a connection is established for direct control of a workflow process.	FMCGRP	No
WorkflowAgentLocatorPolicy	Specifies local or remote connection to the WebSphere MQ Workflow server. LOC The connector connects to the server running on the local machine.OSA The connector connects remotely to the WebSphere MQ Workflow server	LOC	No
WorkflowAgentName	The name of the WebSphere MQ Workflow CORBA agent.		No

ApplicationPassword

Password used with UserID to log in to WebSphere MQ.

Default = password.

If the ApplicationPassword is left blank or removed, the connector uses the default password provided by WebSphere MQ Workflow.

ApplicationUserID

Passed by the connector to the WebSphere MQ Workflow server to authorize a connection (for the Java direct-binding API). This property is also used:

- To authenticate both message delivery via WebSphere MQ Workflow and WebSphere MQ Workflow process execution.
- By the connector during request operations to the WebSphere MQ Workflow application if the user fails to specify attribute UserId in the meta-object (for the XML API).

If ApplicationUserID is left blank or removed, the connector uses the default user ID provided by WebSphere MQ Workflow.

You can specify the user ID in this property or in the meta-object attribute UserID. In either case, ApplicationUserID must be:

- defined in your WebSphere MQ Workflow applications and have all necessary authorizations (for example, to execute the specified workflow)
- defined in your local operating system
- a member of group mqm on your local machine so that WebSphere MQ messages can be sent under its authority.

Note: ApplicationUserID is not specified in the MCA properties for the WebSphere MQ server connection channel used by WebSphere MQ Workflow. By default WebSphere MQ Workflow specifies user fmc for this property, which causes all messages exchanged between the adapter and the WebSphere MQ Workflow application to be sent under the authority of user fmc. Clear this value in your WebSphere MQ server connection channel properties so that messages can be sent by the ApplicationUserID that you specify in this connector-specific property.

Default=ADMIN.

ArchiveQueue
Queue to which copies of successfully processed messages are sent.

Default = `MQWFCONN.ARCHIVE`

MQSeriesChannel
WebSphere MQ Workflow server connector channel through which the connector communicates with WebSphere MQ.

Default=FMCQM.CL.TCP

If the `Channel` is left blank or removed, the connector uses the default server channel provided by WebSphere MQ Workflow.

DataHandlerClassName
Data handler class to use when converting messages to and from business objects.

Default = `com.crossworlds.DataHandlers.text.xml`

DataHandlerConfigMO
Meta-object passed to data handler to provide configuration information.

Default = `MO_DataHandler_Default`

DataHandlerMimeType
Allows you to request a data handler based on a particular MIME type.

Default = `text/xml`

ErrorQueue
Queue to which messages that could not be processed are sent.

Default = `MQWFCONN.ERROR`

MQSeriesHostName
The name of the server hosting WebSphere MQ Workflow.

Default = blank

InDoubtEvents
Specifies how to handle in-progress events that are not fully processed due to unexpected connector shutdown. Choose one of four actions to take if events are found in the in-progress queue during initialization:

- `FailOnStartup.` Log an error and immediately shut down.
- `Reprocess.` Process the remaining events first, then process messages in the input queue.
- `Ignore.` Disregard any messages in the in-progress queue.
- `LogError.` Log an error but do not shut down

Default = `Reprocess`

InputQueue
Message queue that is polled by the connector for new messages.

Default = `CWLDINPUTQ`

InProgressQueue
In-progress event queue.

Default = `MQWFCONN.IN_PROGRESS`

JavaCorbaApi
Enabling the Java CORBA APIs is required for use with WebSphere MQ Workflow 3.2.2. If false, the connector supports the XML APIs for use with WebSphere MQ Workflow 3.3.2 and higher. If you are using WebSphere MQ Workflow 3.4 or later, then this property must be `false`.

Default = `false`

MQSeriesCCSID
The CCSID used to connect to the queue manager for WebSphere MQ Workflow. This value should match the CCSID property of the queue manager for WebSphere MQ Workflow.

Default = blank (if left blank, it is regarded as 819)

You may need to change the CCSID to support selected characters. When you do, you must change the CCSID connector specific property as well as the CCSID of the WebSphere MQ Workflow queue.

Changing the MQSeries CCSID connector property:

1. Double-click MQWF connector in System Manager. The Connector Designer -- MQWorkflowConnector opens.
2. Click the Application Config Properties tab.
3. Enter a new value (such as "943") in the MQSeriesCCSID property.
4. Restart the connector.
5. Restart ICS (recommended).

Changing the MQSeries CCSID queue property:

1. Run RUMMQSC FMCQM at a command prompt.
2. Enter `ALTER QMGR CCSID` (*new_value*) and press Enter.
3. Enter `END` and press Enter.

OutputQueue
Queue to issue requests to MQSeries Workflow.

Default = FMC.FMCGRP.EXE.XML

PollQuantity
Number of messages to retrieve from the input queue.

Default =1

MQSeriesPort
Port established for the MQSeries (WebSphere MQ) listener.

Default = 14000

ReplyToQueue
Queue to which response messages are delivered when the connector issues requests.

Note: When sending messages to the WebSphere MQ Workflow application, the connector populates the ReplyToQueue field in the header of the outbound message regardless of whether the connector expects a response. This helps identify issues when invalid business data is sent to the MQ Workflow application.

Default = `MQWFCONN.REPLYTO`

UnsubscribedQueue

Queue to which messages that are not subscribed are sent. The connector delivers a message to this queue property if:

- The connector retrieves a message that does not have a format of either MQSTR or FMCXML.
- The connector retrieves a `WfMessage` but the message name is not of type `ActivityImplInvoke` or `General Error`.
- The connector cannot find a top-level business object for the data structure when processing subscription delivery.

Default = `MQWFCONN.UNSUBSCRIBE`

MQSeriesQueueManager

The queue manager for WebSphere MQ Workflow.

Default = `FMC` (if left blank, the default queue manager is used)

BOPrefix

The name of the data structure is appended to this prefix for subscription deliveries. The data structure determines the name of the top-level business object for the transaction.

Default = MQWF_

WorkflowSystemName

The name of the WebSphere MQ Workflow system to which a connection will be established when direct control of a workflow process is required by the connector.

Default = `FMCSYS`

WorkflowSystemGroup

The name of the WebSphere MQ Workflow system group to which a connection will be established when direct control of a workflow process is required by the connector.

Default = `FMCGRP`

WorkflowAgentLocatorPolicy

Specifies how the connector establishes a connection to the WebSphere MQ Workflow server identified by properties *WorkflowSystemName* and *WorkflowSystemGroup*. Possible values are as follows:

- LOC. The connector connects to the WebSphere MQ Workflow server running on the local machine.
- OSA. The connector connects remotely to the WebSphere MQ Workflow server using IBM Java Object Request Broker (ORB). WebSphere MQ Workflow must be configured to support IBM Java ORB clients. See the *WebSphere MQ Workflow Installation Guide* and *WebSphere MQ Workflow Programming Guide* for more information.

Default = LOC

Note: In order to support client connections with IBM Java ORB, the `start_MQWorkflow.bat` (Windows) or `start_MQWorkflow.sh` (UNIX) files need to be modified. Open the appropriate `start_MQWorkflow` file and scroll down until you see a comment beginning with Step 3... all of the lines that follow and adjust paths as indicated in the directions provided. This ensures that the correct IBM Java ORB libraries are loaded and used by the WebSphere MQ Workflow client libraries during initialization. This modification does not affect your communication with InterChange Server.

WorkflowAgentName

The name of the WebSphere MQ Workflow CORBA agent.

Default = none

Enabling guaranteed event delivery

You can configure the guaranteed-event-delivery feature for a JMS-enabled connector in one of the following ways:

- If the connector uses a JMS event store (implemented as a JMS source queue), the connector framework can manage the JMS event store. For more information, see "Guaranteed event delivery for connectors with JMS event stores."
- If the connector uses a non-JMS event store (for example, implemented as a JDBC table, Email mailbox, or flat files), the connector framework can use a JMS monitor queue to ensure that no duplicate events occur. For more information, see "Guaranteed event delivery for connectors with non-JMS event stores" on page 29.

Guaranteed event delivery for connectors with JMS event stores

If the JMS-enabled connector uses JMS queues to implement its event store, the connector framework can act as a "container" and manage the JMS event store (the JMS source queue). In a single JMS transaction, the connector can remove a message from a source queue and place it on the destination queue. This section provides the following information about use of the guaranteed-event-delivery feature for a JMS-enabled connector that has a JMS event store:

- "Enabling the feature for connectors with JMS event stores"
- "Effect on event polling" on page 29

Enabling the feature for connectors with JMS event stores

To enable the guaranteed-event-delivery feature for a JMS-enabled connector that has a JMS event store, set the connector configuration properties to values shown in Table 5.

Table 5. Guaranteed-event-delivery connector properties for a connector with a JMS event store

Connector property	Value
`DeliveryTransport`	`JMS`
`ContainerManagedEvents`	`JMS`
`PollQuantity`	The number of events to processing in a single poll of the event store

Table 5. Guaranteed-event-delivery connector properties for a connector with a JMS event store (continued)

Connector property	Value
SourceQueue	Name of the JMS source queue (event store) which the connector framework polls and from which it retrieves events for processing **Note:** The source queue and other JMS queues should be part of the same queue manager. If the connector's application generates events that are stored in a different queue manager, you must define a remote queue definition on the remote queue manager. WebSphere MQ can then transfer the events from the remote queue to the queue manager that the JMS-enabled connector uses for transmission to the integration broker. For information on how to configure a remote queue definition, see your IBM WebSphere MQ documentation.

In addition to configuring the connector, you must also configure the data handler that converts between the event in the JMS store and a business object. This data-handler information consists of the connector configuration properties that Table 6 summarizes.

Table 6. Data-handler properties for guaranteed event delivery

Data-handler property	Value	Required?
MimeType	The MIME type that the data handler handles. This MIME type identifies which data handler to call.	Yes
DHClass	The full name of the Java class that implements the data handler	Yes
DataHandlerConfigMOName	The name of the top-level meta-object that associates MIME types and their data handlers	Optional

Note: The data-handler configuration properties reside in the connector configuration file with the other connector configuration properties.

If you configure a connector that has a JMS event store to use guaranteed event delivery, you must set the connector properties as described in Table 5 *and* Table 6. To set these connector configuration properties, use the Connector Configurator tool. Connector Configurator displays the connector properties in Table 5 on its Standard Properties tab. It displays the connector properties in Table 6 on its Data Handler tab.

Note: Connector Configurator activates the fields on its Data Handler tab only when the DeliveryTransport connector configuration property is set to JMS and ContainerManagedEvents is set to JMS.

For information on Connector Configurator, see Appendix B, "Connector Configurator," on page 91.

Effect on event polling

If a connector uses guaranteed event delivery by setting `ContainedManagedEvents` to `JMS`, it behaves slightly differently from a connector that does not use this feature. To provide container-managed events, the connector framework takes the following steps to poll the event store:

1. Start a JMS transaction.
2. Read a JMS message from the event store.

   The event store is implemented as a JMS source queue. The JMS message contains an event record. The name of the JMS source queue is obtained from the `SourceQueue` connector configuration property.
3. Call the data handler to convert the event to a business object.

   The connector framework calls the data handler that has been configured with the properties in Table 6 on page 28.
4. When WebSphere MQ Integrator Broker is the integration broker, convert the business object to a message based on the configured wire format (XML).
5. Send the resulting message to the JMS destination queue.
   If you are using the WebSphere ICS integration broker, the message sent to the JMS destination queue is the business object. If you are using WebSphere MQ Integrator broker, the message sent to the JMS destination queue is an XML message (which the data handler generated).
6. Commit the JMS transaction.

   When the JMS transaction commits, the message is written to the JMS destination queue and removed from the JMS source queue in the same transaction.
7. Repeat step 1 through 6 in a loop. The `PollQuantity` connector property determines the number of repetitions in this loop.

Important: A connector that sets the `ContainerManagedEvents` property is set to `JMS` does *not* call the `pollForEvents()` method to perform event polling. If the connector's base class includes a `pollForEvents()` method, this method is *not* invoked.

Guaranteed event delivery for connectors with non-JMS event stores

If the JMS-enabled connector uses a non-JMS solution to implement its event store (such as a JDBC event table, Email mailbox, or flat files), the connector framework can use duplicate event elimination to ensure that duplicate events do not occur. This section provides the following information about use of the guaranteed-event-delivery feature with a JMS-enabled connector that has a non-JMS event store:

- "Enabling the feature for connectors with non-JMS event stores"
- "Effect on event polling"

Enabling the feature for connectors with non-JMS event stores: To enable the guaranteed-event-delivery feature for a JMS-enabled connector that has a non-JMS event store, you must set the connector configuration properties to values shown in Table 7.

Table 7. Guaranteed-event-delivery connector properties for a connector with a non-JMS event store

Connector property	Value
`DeliveryTransport`	`JMS`

Table 7. Guaranteed-event-delivery connector properties for a connector with a non-JMS event store (continued)

Connector property	Value
DuplicateEventElimination	true
MonitorQueue	Name of the JMS monitor queue, in which the connector framework stores the ObjectEventId of processed business objects

If you configure a connector to use guaranteed event delivery, you must set the connector properties as described in Table 7. To set these connector configuration properties, use the Connector Configurator tool. It displays these connector properties on its Standard Properties tab. For information on Connector Configurator, see Appendix B, "Connector Configurator," on page 91.

Effect on event polling: If a connector uses guaranteed event delivery by setting DuplicateEventElimination to true, it behaves slightly differently from a connector that does not use this feature. To provide the duplicate event elimination, the connector framework uses a JMS monitor queue to track a business object. The name of the JMS monitor queue is obtained from the MonitorQueue connector configuration property.

After the connector framework receives the business object from the application-specific component (through a call to gotApplEvent() in the pollForEvents() method), it must determine if the current business object (received from gotApplEvents()) represents a duplicate event. To make this determination, the connector framework retrieves the business object from the JMS monitor queue and compares its ObjectEventId with the ObjectEventId of the current business object:

- If these two ObjectEventIds are the same, the current business object represents a duplicate event. In this case, the connector framework ignores the event that the current business object represents; it does *not* send this event to the integration broker.
- If these ObjectEventIds are *not* the same, the business object does *not* represent a duplicate event. In this case, the connector framework copies the current business object to the JMS monitor queue and then delivers it to the JMS delivery queue, all as part of the same JMS transaction. The name of the JMS delivery queue is obtained from the DeliveryQueue connector configuration property. Control returns to the connector's pollForEvents() method, after the call to the gotApplEvent() method.

For a JMS-enabled connector to support duplicate event elimination, you must make sure that the connector's pollForEvents() method includes the following steps:

- When you create a business object from an event record retrieved from the non-JMS event store, save the event record's unique event identifier as the business object's ObjectEventId attribute.

  The application generates this event identifier to uniquely identify the event record in the event store. If the connector goes down after the event has been sent to the integration broker but before this event record's status can be changed, this event record remains in the event store with an In-Progress status. When the connector comes back up, it should recover any In-Progress events. When the connector resumes polling, it generates a business object for the event record that still remains in the event store. However, because both the business

object that was already sent and the new one have the same event record as their ObjectEventIds, the connector framework can recognize the new business object as a duplicate and not send it to the integration broker.

- During connector recovery, make sure that you process In-Progress events *before* the connector begins polling for new events.

  Unless the connector changes any In-Progress events to Ready-for-Poll status when it starts up, the polling method does not pick up the event record for reprocessing.

Top-level business object and content configuration

metadata is embedded along with business object data in the WfMessage structure of WebSphere MQ Workflow. This structure is the basis for all requests and responses between the connector and WebSphere MQ Workflow using the XML message API. The structure of all messages is shown in Figure 7:

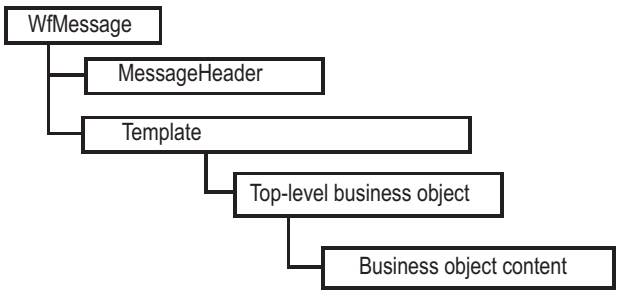

Figure 7. WebSphere MQ Workflow message structure

Commands and return values in the XML API are encompassed by "templates". These templates provide the structures necessary to specify entire commands to WebSphere MQ Workflow as well as to contain the results. The type of template changes depending on the action requested and, in most cases, the business content is contained in a child element of the template. Identifying the business content requires that the connector recognize each template specifically. As the names of the templates differ, so do the names of the child elements.

The connector can process three templates and their associated response structures:

ProcessTemplateExecute
: Sent by the connector to the WebSphere MQ Workflow server to execute a process either synchronously or asynchronously. If the process is asynchronously executed, no response is issued by WebSphere MQ Workflow. If the process is executed synchronously, a response is returned only after the workflow process has completed. A business object representing the workflow input data structure is contained in child element ProcInstInputData.

ProcessTemplateExecuteResponse
: Sent by WebSphere MQ Workflow in response to a synchronous request issued by the connector. The business object that results from the workflow process is contained in child element ProcInstOutputData. A process instance identifier (PID) is returned, although it is no longer active and cannot be used to control the workflow further.

ProcessTemplateCreateAndStartInstance
: Sent by the connector to the WebSphere MQ Workflow server to

execute a process asynchronously. Unlike in the ProcessTemplateExecute template, a response is issued immediately to the connector containing the active PID (instead of a business object). This PID can later be used to control the workflow process. A business object representing the data structure destined for the workflow is contained in child element ProcInstInputData.

ProcessTemplateCreateAndStartInstanceResponse
Sent by WebSphere MQ Workflow in response to a request sent by the connector. A PID is returned without a business object (because the workflow is assumed to be executing asynchronously).

ActivityImplInvoke
Sent by WebSphere MQ Workflow to the connector to request that business content be posted to InterChange Server. The business object is contained by child element ProgramInputData. WebSphere MQ Workflow may include an additional child element ProgramOutputDataDefault that contains default values for the business content returned to the workflow for synchronous requests.

ActivityImplInvokeResponse
Returned by the connector to WebSphere MQ Workflow to complete synchronous requests processed during event polling. The business object returned by the collaboration is added to child element ProgramOutputData.

Depending on the structure of the processed template, the connector must either retrieve or add business content from one of the following XML child elements:

- ProcInstInputData
- ProcInstOutputData
- ProgramInputData
- ProgramOutputData

Meta-object configuration

The business object itself consists of a top-level business object that holds one or more child objects representing the data structure. This means each business object holds some child objects that contain metadata and one or more child objects that contain the actual business content.

The top-level business object is a wrapper for all objects that will be exchanged. As a wrapper, the top-level business object holds no business data, but rather provides the application-specific text and name-value pairs needed to build the objects required to initiate and complete the exchange. The name-value pairs indicate child attributes that specify metadata needed to create the requested or required business objects.

Figure 8 shows the top-level business object and child object relationships when:

- the connector polls for events and receives a request from WebSphere MQ Workflow
- WebSphere MQ Workflow mode is asynchronous, or WebSphere MQ Workflow mode is synchronous and the collaboration name is specified in the WebSphere MQ Workflow command line parameters (see Figure 20 on page 56)

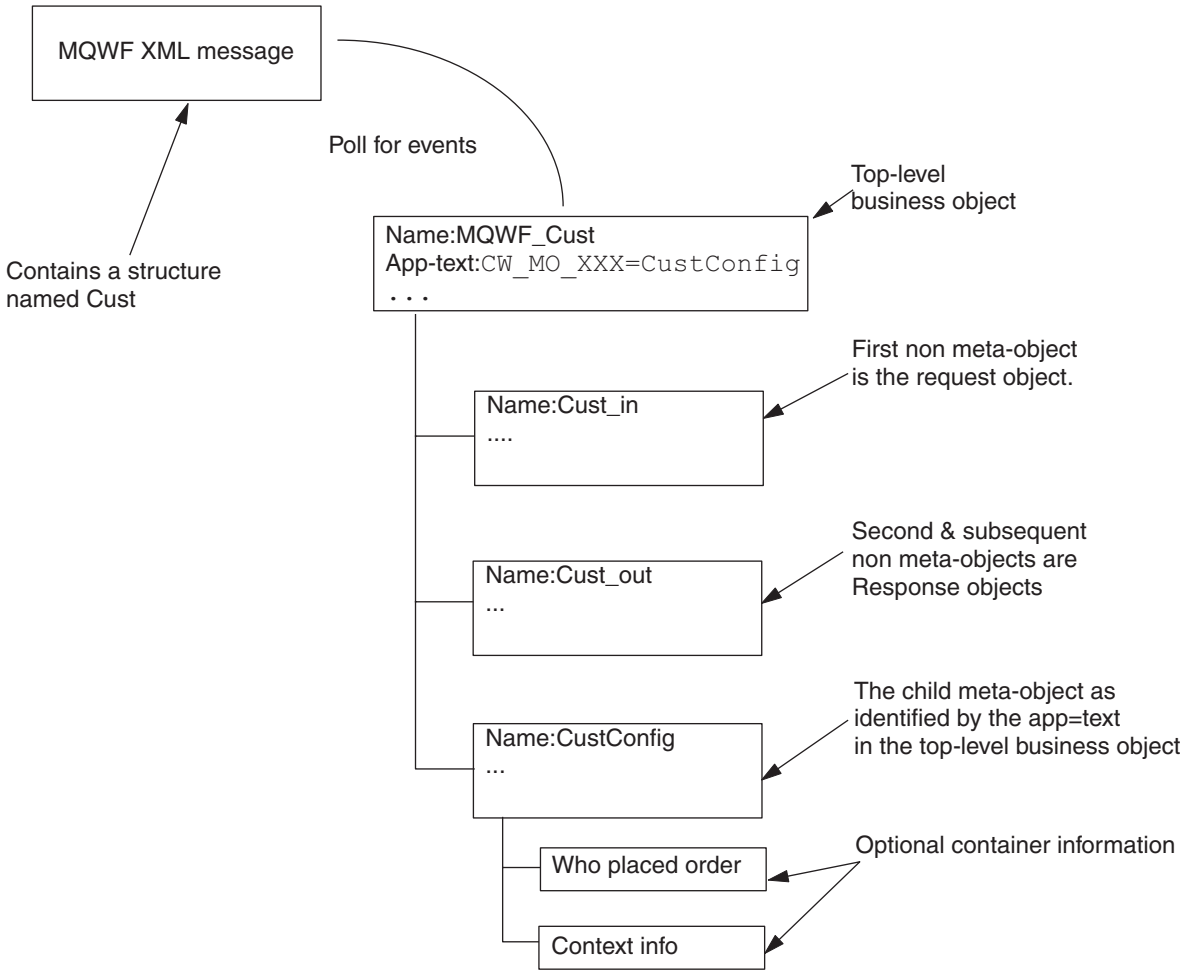

Figure 8. Top-level business object and child (business and meta-objects) in poll for events scenario

Figure 9 shows the top-level business object and child object relationships when:

- the connector polls for events and receives a request from WebSphere MQ Workflow
- WebSphere MQ Workflow mode is synchronous and the collaboration name is not specified in the WebSphere MQ Workflow command line parameters (see Figure 20 on page 56)

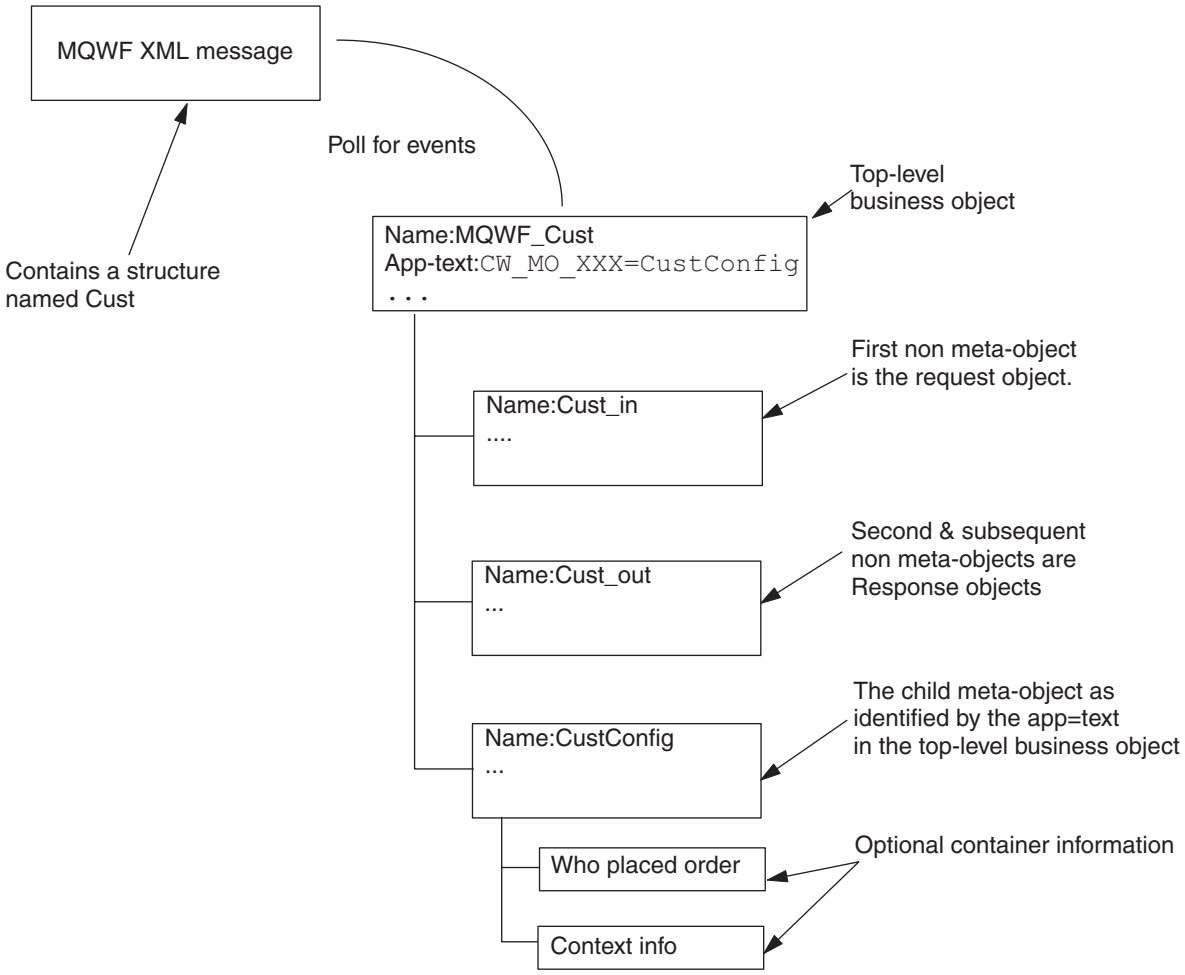

Figure 9. Top-level business object and child meta-objects in poll for events scenario

Figure 10 shows the top-level business object and child object relationships when:

- the connector issues a request on behalf of a collaboration to WebSphere MQ Workflow
- the `cw_mo_wfactivityresponse` tag is not specified in the application-specific information (app-text) of the top-level business object

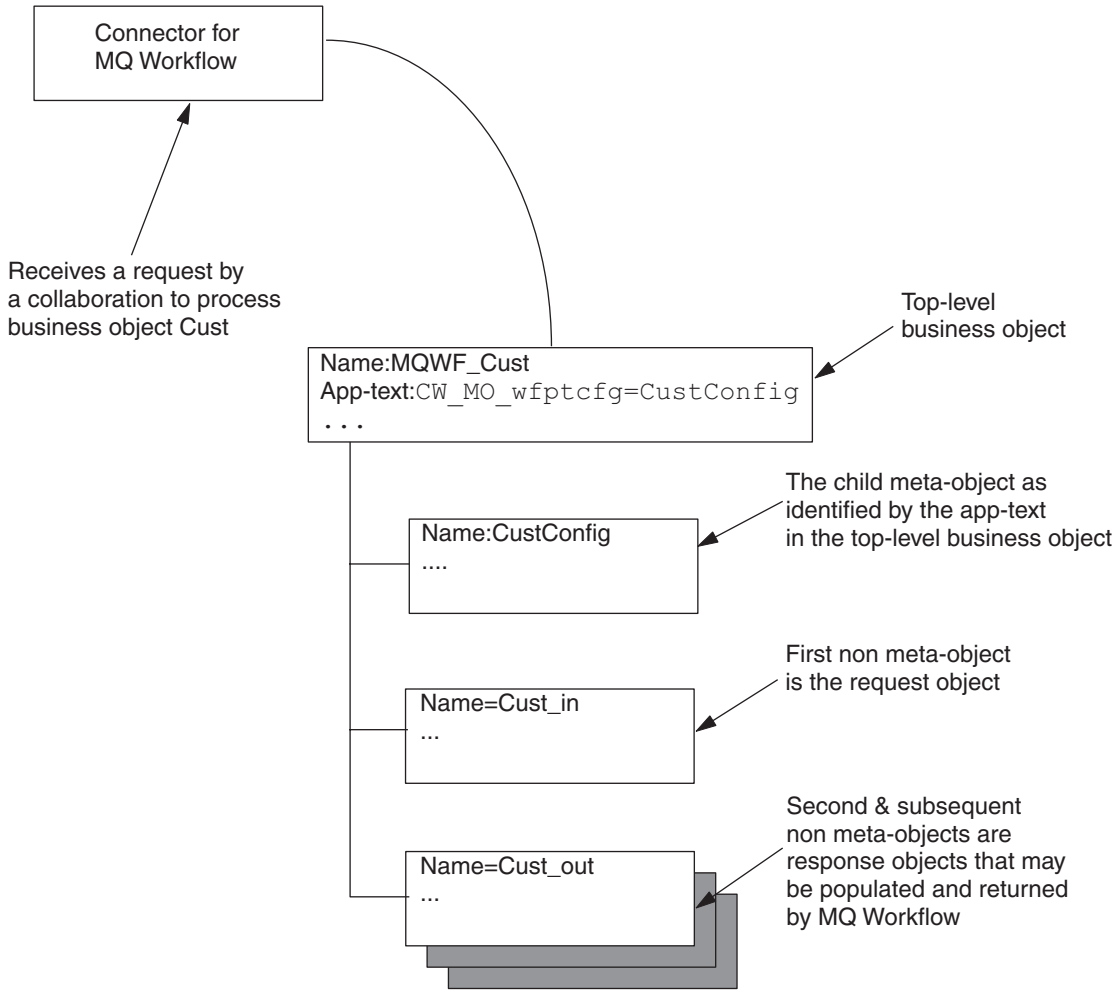

Figure 10. Top-level business object and child (business and meta) objects in request processing scenario

Figure 11 shows the top-level business object and child object relationships when:

- the connector issues a request on behalf of a collaboration to WebSphere MQ Workflow
- the `cw_mo_wfactivityresponse` tag is specified in the application-specific information (app-text) of the top-level business object (see

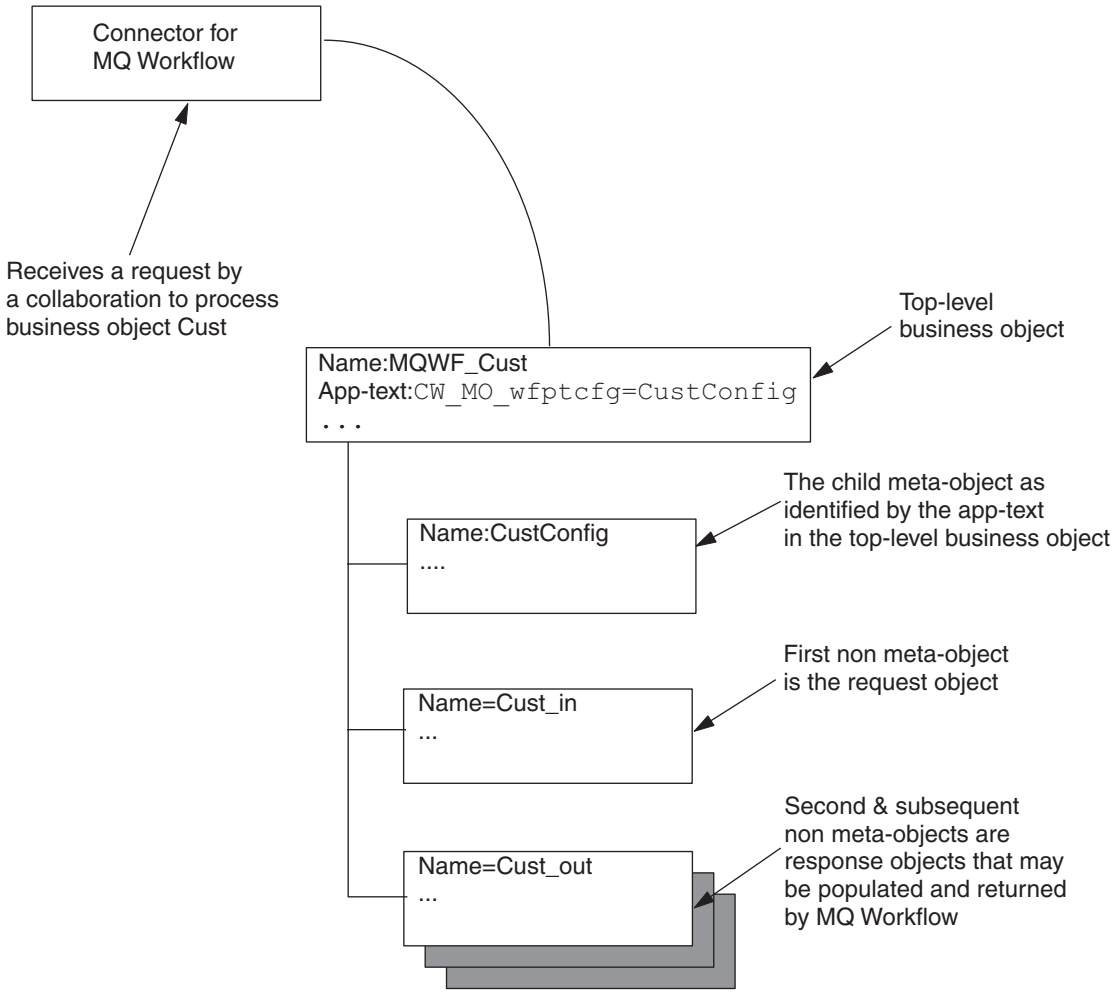

Figure 11. Top-level business object and child (business and meta-) objects in request processing scenario

The WebSphere MQ Workflow process can have different input and output data structures, but transactions involving a collaboration and a connector can involve only one object type. To get around this limitation, some top-level business objects are constructed that have, as children, a request object and one or more response objects. Specifically, the name of the data structure appended to the *<boprefix>* configuration property determines which top-level business object is created. The first (non metadata related) child object in this top-level business object is populated with the data structure.

To distinguish child meta-objects from business-content child objects, the application-specific information for the top-level business object must include a tag as follows:

`cw_mo_`*`tag=child meta-object_attribute_name`*

Note: Multiple metadata tags should be delimited by a semi-colon. White space surrounding delimiters is ignored. (For example: `cw_mo_foo = bar` is equivalent to `cw_mo_foo=bar`)

where `tag` = one of the following:

- `wfptcfg` WebSphere MQ Workflow process template metadata— see "MO_MQWorkflow_ProcessTemplateConfig" on page 37

- `wfcontainer` WebSphere MQ Workflow container information—see "MO_MQWorkflow_ContainerInfo" on page 38
- `wfpid` WebSphere MQ Workflow process instance information—see "MO_MQWorkflow_ProcessInstance" on page 43
- `wfactivityresponse` WebSphere MQ Workflow activity information—see "MO_MQWorkflow_ActivityResponse" on page 45

All application-specific information beginning with `cw_mo_` is reserved for configuration or dynamic metadata. Accordingly, when a connector agent receives a business object, it can immediately determine if any runtime metadata has been included for the business object by simply checking the application-specific information of the business object itself. Data handlers, too, check the application-specific information at the business object level to determine which child objects to include or exclude in the serialization or de-serialization process.

For example, consider a top-level business object named `WfRequest_MyCustomer` that needs a configuration meta-object. You can specify an object configuration attribute of type `MyConfig`. To allow the connector to recognize the meta-object and keep the data handler from incorporating "MyConfig" when serializing the parent object, you add application-specific information to the `WRequest_MyCustomer` object in the form of the tag `cw_mo_wfptcfg=MyConfig`.

When constructing a request or response for WebSphere MQ Workflow, the connector uses templates to construct business objects.

MO_MQWorkflow_ProcessTemplateConfig

To provide information on the WebSphere MQ Workflow process to be created and executed, the connector requires that a meta-object be included in the top-level business object. This meta-object includes information regarding the process template to use, whether a response is required, whether WebSphere MQ Workflow must wait until the process is compete before returning a result, and more. By storing this information in a meta-object, the connector can dynamically configure application-specific information for the requested WebSphere MQ Workflow process. This meta-object (or its equivalent) is required for all collaboration requests.

The connector reads the application-specific information of the top-level business object and looks for a name value pair:

```
cw_mo_wfptcfg=xxx
```

where `xxx` is the name of the child attribute that specifies the metadata. Table 8 shows the attribute names and descriptions.

Table 8. MO_MQWorkflow_ProcessTemplateConfig meta-object attributes

Attribute name	Description	Accepted values
`ProcessTemplateName`(Required)	Name of WebSphere MQ Workflow template to execute. Default = none	Any
`ProcessInstanceName`	Name of WebSphere MQ Workflow instance to execute. Does not apply in asynchronous execution mode. Default = If left blank, a new instance of the process template is created.	Any

Table 8. MO_MQWorkflow_ProcessTemplateConfig meta-object attributes (continued)

Attribute name	Description	Accepted values
KeepName	Flag indicating whether a process should be discarded after use. Default = false	true or false
UserID	Identifies a user with authorization to execute the process. This is the same as the connector-specific property, ApplicationUserID. For important restrictions on this attribute, see "ApplicationUserID" on page 23. Default = Value of connector configuration property ApplicationUserID	Any
ResponseTimeout	Amount of time (in ms) to wait for a response from WebSphere MQ Workflow. A positive value requires the connector to wait for a response. A negative value causes the connector to return successfully once a request is issued to the WebSphere MQ Workflow input queue. Default = -1	Integer
TimeoutFatal	If a response is not received by WebSphere MQ Workflow, the connector returns BON_APPRESPONSETIMEOUT to InterChange Server and terminates the connector agent. Default = false (does not apply if ResponseTimeout is less than 0)	true or false
ExecutionMode	Determines whether a process executes asynchronously or synchronously in relation to the collaboration. When this mode is Asynchronous, a new instance of the process template is created and executed. A PID is returned to the collaboration for tracking purposes. When this mode is Synchronous, an instance (either existing or new) of a process template is executed. The resulting business object is returned to the collaboration once the workflow process is completed. Default = Synchronous	Asynchrounous or Synchronous

MO_MQWorkflow_ContainerInfo

Activity invoking (ActivityImplInvoke)messages issued by WebSphere MQ Workflow optionally can hold container information in addition to a business object. This container information can be mapped to the child meta-object MO_MQWorkflow_ContainerInfo (if defined) and published to subscribing collaborations. It includes information provided by WebSphere MQ Workflow regarding the conditions and environments in which the process originated.

Note: The MO_MQWorkflow_ContainerInfo metadata is for informational purposes only. A collaboration may choose to ignore it or take different actions based on the process model or role of the user initiating the requested process.

The connector reads the application-specific information of the top-level business object and looks for the name-value pair:

```
cw_mo_wfcontainer=XXX
```

where XXX is the name of the child attribute to populate with the top-level business object information. This information has no value to the connector and is not used for processing business objects. Passing this information to the connector as part of a business object has no effect.

Here is a the MO_MQWorkflow_ContainerInfo definition.

```
[ReposCopy]

     Version = 3.1.0
[End]
[BusinessObjectDefinition]
Name = MO_MQWorkflow_ProcessInfo
Version = 1.0.0


  [Attribute]
  Name = Role
  Type = String
  Cardinality = 1
  MaxLength = 255
  IsKey = true
  IsForeignKey = false
  IsRequired = false
  IsRequiredServerBound = false
  [End]
  [Attribute]
  Name = Organization
  Type = String
  Cardinality = 1
  MaxLength = 255
  IsKey = false
  IsForeignKey = false
  IsRequired = false
  IsRequiredServerBound = false
  [End]
  [Attribute]
  Name = ProcessAdministrator
  Type = String
  Cardinality = 1
  MaxLength = 255
  IsKey = false
  IsForeignKey = false
  IsRequired = false
  IsRequiredServerBound = false
  [End]
  [Attribute]
  Name = Duration
  Type = String
  Cardinality = 1
  MaxLength = 255
  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]
[BusinessObjectDefinition]
Name = MO_MQWorkflow_ActivityInfo
Version = 1.0.0

    [Attribute]
    Name = Priority
    Type = String
    Cardinality = 1
    MaxLength = 255
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = MembersOfRoles
    Type = String
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = CoordinatorOfRole
    Type = String
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = Organization
    Type = String
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = OrganizationType
    Type = String
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = LowerLevel
    Type = String
    Cardinality = 1
```

```
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = UpperLevel
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = People
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = PersonToNotify
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Duration
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Duration2
Type = String
Cardinality = 1
MaxLength = 255
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]
[BusinessObjectDefinition]
Name = MO_MQWorkflow_ContainerInfo
Version = 1.0.0

    [Attribute]
    Name = PROCESS_INFO
    Type = MO_MQWorkflow_ProcessInfo
    ContainedObjectVersion = 1.0.0
    Relationship = Containment
    Cardinality = 1
    MaxLength = 1
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = ACTIVITY_INFO
    Type = MO_MQWorkflow_ActivityInfo
    ContainedObjectVersion = 1.0.0
    Relationship = Containment
    Cardinality = 1
    MaxLength = 1
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = ACTIVITY
    Type = String
    Cardinality = 1
    MaxLength = 1
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = PROCESS
    Type = String
    Cardinality = 1
    MaxLength = 1
    IsKey = true
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]
    [Attribute]
    Name = PROCESS_MODEL
    Type = String
    Cardinality = 1
```

```
    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]
```

MO_MQWorkflow_ProcessInstance

Optionally, a process instance ID (PID) is returned by WebSphere MQ Workflow in response to a process execution. Upon successful creation or execution of a workflow process, the connector populates this object with details of the process. If the process is executing in parallel to the initiating collaboration, the collaboration can use the PID to control the process instance (if the process was executed asynchronously to the collaboration).

The connector reads the application-specific information of the parent business object and looks for a name-value pair:

```
cw_mo_wfpid

=XXX
```

where `XXX` is the name of the child attribute that should contain the process instance metadata. The object must conform to the requirements set forth by the XML data handler. Furthermore, the names of the attributes in the child object have semantic value to the connector. The application-specific information for the object must include "ProcessInstance".

It is highly recommended that all custom objects be derived from this sample.

Table 9. MO_MQWorkflow_ProcessTemplateInstance meta-object attributes

Attribute name	Description	Accepted values
`ProcInstID`	Primary key identifying the process instance	Any
`ProcessInstName`	See the WebSphere MQ Workflow programming guide.	

Table 9. MO_MQWorkflow_ProcessTemplateInstance meta-object attributes (continued)

Attribute name	Description	Accepted values
ProcInstParentName	See the WebSphere MQ Workflow programming guide.	
ProcInstTopLevelName	See the WebSphere MQ Workflow programming guide.	Any
ProcInstDescription	See the WebSphere MQ Workflow programming guide.	Integer
ProcInstState	State of Process	SuspendedResumedTerminated
LastStateChangeTime	See the WebSphere MQ Workflow programming guide.	Asynchronous or Synchronous
LastModificationTime	See the WebSphere MQ Workflow programming guide.	
ProcTempID	See the WebSphere MQ Workflow programming guide.	
ProcTempIName	See the WebSphere MQ Workflow programming guide.	
Icon	See the WebSphere MQ Workflow programming guide.	
Category	See the WebSphere MQ Workflow programming guide.	

MO_MQWorkflow_ActivityRequest

When an WebSphere MQ Workflow process instance issues an activity-invoking (ActivityImplInvoke) message and the following conditions are met:

- the WebSphere MQ Workflow mode is synchronous (see Figure 22 on page 58)
- the collaboration name is not specified in the WebSphere MQ Workflow command line parameters (see Figure 20 on page 56)

the connector extracts the activity information from the ActivityImplInvoke message. This activity information is mapped to the child meta-object MO_MQWorkflow_ActivityRequest and then published to subscribing collaborations.

The connector reads the application-specific information of the top-level business object and looks for the following name-value pair:

```
cw_mo_wfactivityrequest

=XXX
```

where XXX is the name of the child attribute to populate with the top-level business object information. This information is neither of value to the connector nor used for processing business objects. Table 10 shows the attribute names and descriptions

Table 10. MO_MQWorkflow_ActivityRequest Attributes

Attribute Name	Description	Accepted Values
ActImplCorrelID	The ID that correlates the ActivityImplInvoke message issued by the process instance with the ActivityImplInvokeResponse message. The collaboration uses ActImplCorrelID to send the response to the connector	Any
Starter	The user ID that initiated the process instance	Any
ProcTemplID	The ID of the process template.	Any
ProgramName	The program name which is called by the process instance	Any
ResponseRequired	Identifies whether the process (or instance) expects the ActivityImplInvokeResponse message	Yes or No or IfError
ExternalProcessContext	The process context of the process instance	Any

MO_MQWorkflow_ActivityResponse

To correlate a response with a request, the connector requires that the top-level business object include a meta-object with appropriate information. This meta-object includes information regarding the ActImplCorrelID of the associated ActivityImplInvoke message, the user ID (Starter) associated with the process instance, and the return code linked to the process instance. This meta-object is required when a collaboration associates an ActivityImplInvokeResponse with the ActivityImplInvoke message.

The connector reads the application-specific information of the top-level business object and looks for the following name value pair:

cw_mo_wfactivityrequest

=XXX

where XXX is the name of the child attribute that specifies the metadata. Table 11 shows the attribute names and descriptions

Table 11. MO_MQWorkflow_ActivityResponse Attributes

Attribute Name	Description	Accepted Values
ActImplCorrelID	ID by which the process instance correlates the response with the request invoked by the associated ActivityImplInvoke message.	Any
Starter	The user ID that initiated the process instance associated with the ActivityImplInvoke message	Any
ReturnCode	The return code issued to the process instance	Integer

Application-Specific Information

The application-specific information at the parent business object level is structured in name-value pair format, separated by semicolons. White space is ignored. For example:

```
cw_mo_wfptcfg=CUST_Config;cw_mo_pid=CUST_IN_Nieman
```

Startup file configuration

Before you start the connector for WebSphere MQ Workflow, you must configure the startup file.

Windows

To complete the configuration of the connector for Windows platforms, you must modify the `start_WebSphereMQWorkflow.bat` file:

1. Open the `start_WebSphereMQWorkflow.bat` file.
2. Scroll to the section beginning with `STEP 1` and specify the location of your WebSphere MQ Java client libraries.
3. Scroll to the section beginning with `STEP 2` and specify the location of your Workflow Java client libraries.
4. If you intend to connect to WebSphere MQ Workflow via IBM Java ORB, scroll to the section beginning with step 3, specify the location of your IBM Java ORB libraries and uncomment these lines.

UNIX

To complete the configuration of the connector for UNIX platforms, you must modify the `start_WebSphereMQWorkflow.sh` file:

1. Open the `start_WebSphereMQWorkflow.sh` file.
2. Scroll to the section beginning with `STEP 1` and specify the location of your WebSphere MQ Java client libraries.
3. Scroll to the section beginning with `STEP 2` and specify the location of your Workflow Java client libraries.
4. If you intend to connect to WebSphere MQ Workflow via IBM Java Object Request Broker (ORB), scroll to the section beginning with step 3, specify the location of your IBM Java ORB libraries and uncomment these lines.

Creating multiple instances 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 46.
3. Create a startup script shortcut (Windows only).
4. Copy the initial connector's shortcut text and change the name of the initial connector (in the command line) to match the name of the new connector instance.

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

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

Starting the connector

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

ProductDir\connectors*connName*

where *connName* identifies the connector. The name of the startup script depends on the operating-system platform, as Table 12 shows.

Table 12. Startup scripts for a connector

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

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

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

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

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

    `connector_manager_`*`connName`* `-start`

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

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

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

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

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

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

Stopping the connector

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

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

    ```
    connector_manager_connName -stop
    ```

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

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

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

Chapter 3. Modifying the WebSphere MQ Workflow application

In the WebSphere MQ Workflow application, nodes are steps in a process that get work done. Each node is associated with a certain activity (for example, order approval) that corresponds to a role (department head).

Nodes can issue requests and respond to other nodes or to applications that are external to WebSphere MQ Workflow. To enable a node to communicate with the connector for WebSphere MQ Workflow, you must first specify a User-Defined Program Execution Server (UPES).

This chapter describes how create a User-Defined Program Execution Server (UPES) that can interact with the connector for WebSphere MQ Workflow. It covers the following topic:

- "Configuring the UPES"

Configuring the UPES

This chapter describes how to define and configure a User-defined Program Execution Server (UPES). Via a UPES, workflow nodes can issue requests to the connector for WebSphere MQ Workflow.

Note: You must have the WebSphere MQ Workflow Buildtime environment installed on your system before configuring a UPES.

1. Start the MQ Workflow Buildtime application and click the Network tab.

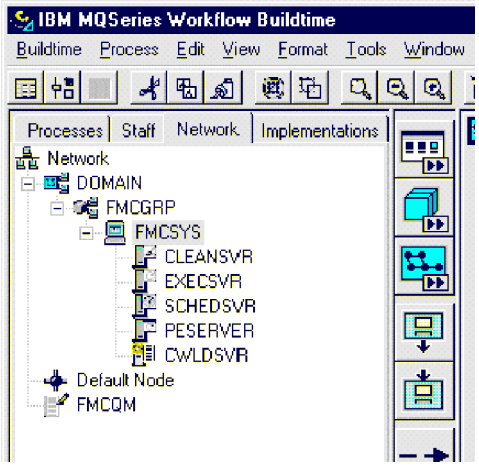

Figure 12. WebSphere MQ Workflow Buildtime: Network view

2. From the menu bar, select System > New User-Defined Program Execution Server.

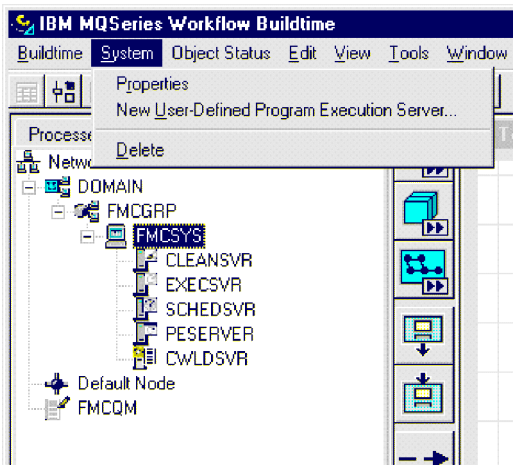

Figure 13. WebSphere MQ Workflow Buildtime: Choosing the new UPES

3. In the dialog box, enter a unique name for the UPES (for example, CWLDSVR).

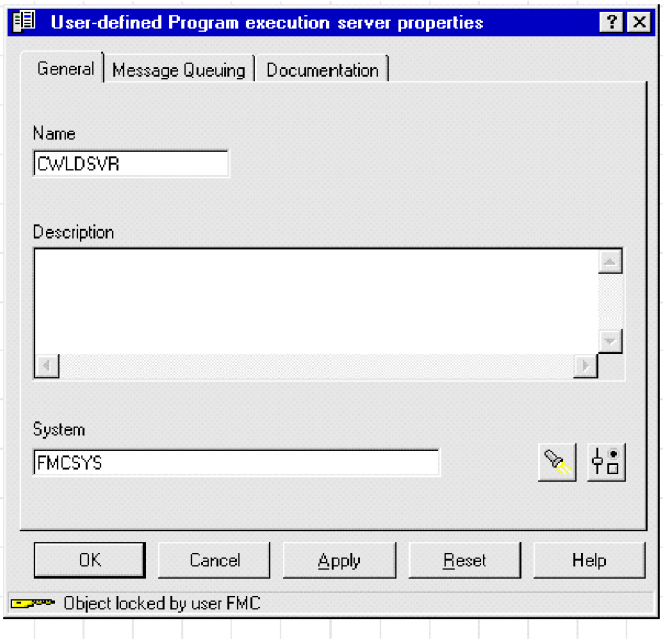

Figure 14. WebSphere MQ Workflow Buildtime: Naming the new UPES

4. Click the Message Queuing tab and enter the names of the input queue and the queue manager for the connector.

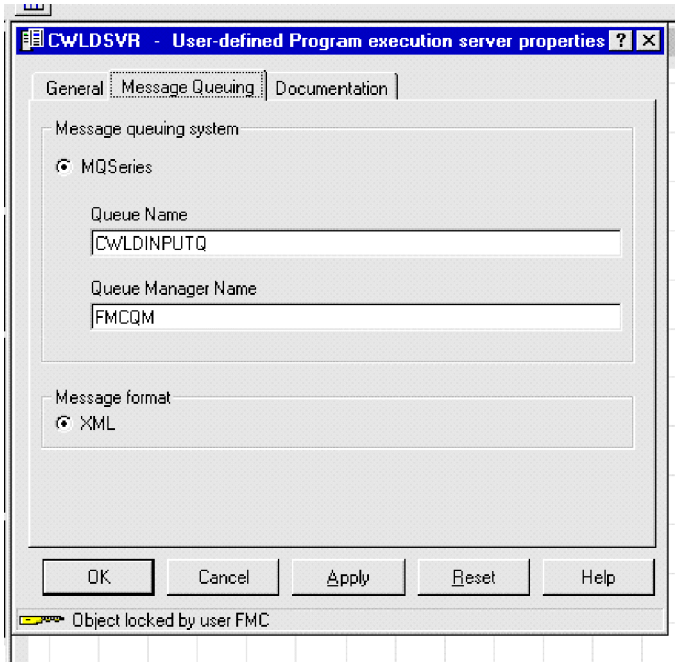

Figure 15. WebSphere MQ Workflow Buildtime: Configuring the message queue

5. Click the Implementations tab.

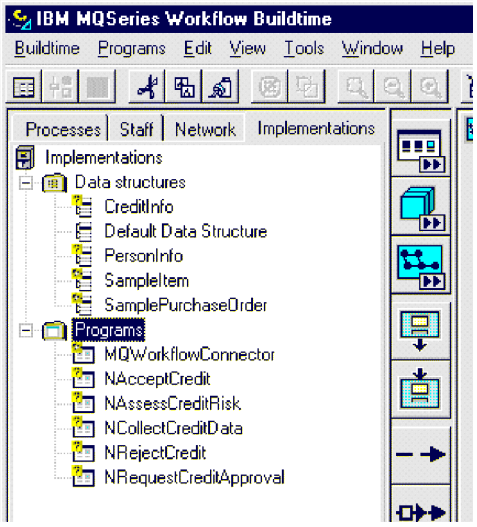

Figure 16. WebSphere MQ Workflow Buildtime: Implementations view

6. Select Programs > New Program from the menu bar.

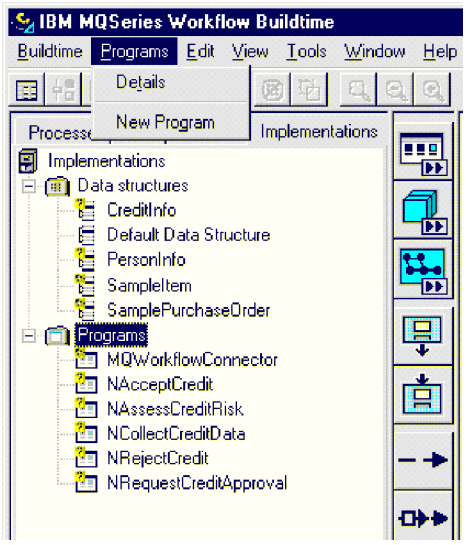

Figure 17. WebSphere MQ Workflow Buildtime: Choosing new program

7. Specify a name for the program. Because a separate UPES program must be defined for each node-to-collaboration relationship, you may want to use the same name as the collaboration.

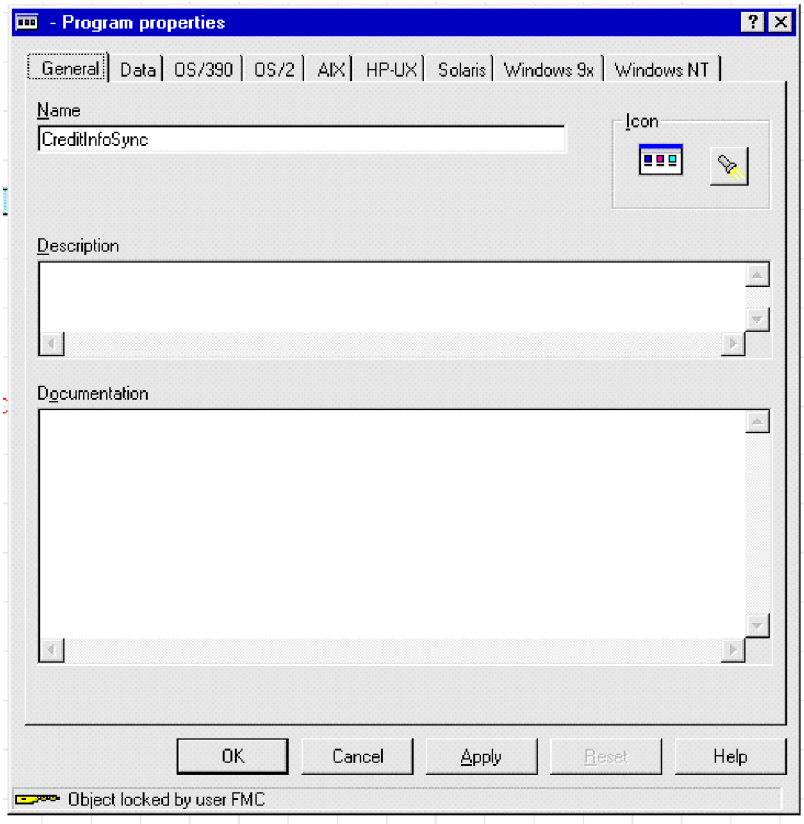

Figure 18. WebSphere MQ Workflow Buildtime: Naming the new program

8. Click the Data tab and specify the data structures to be accepted by the program or collaboration. Ensure that box “Program can run unattended” is checked.

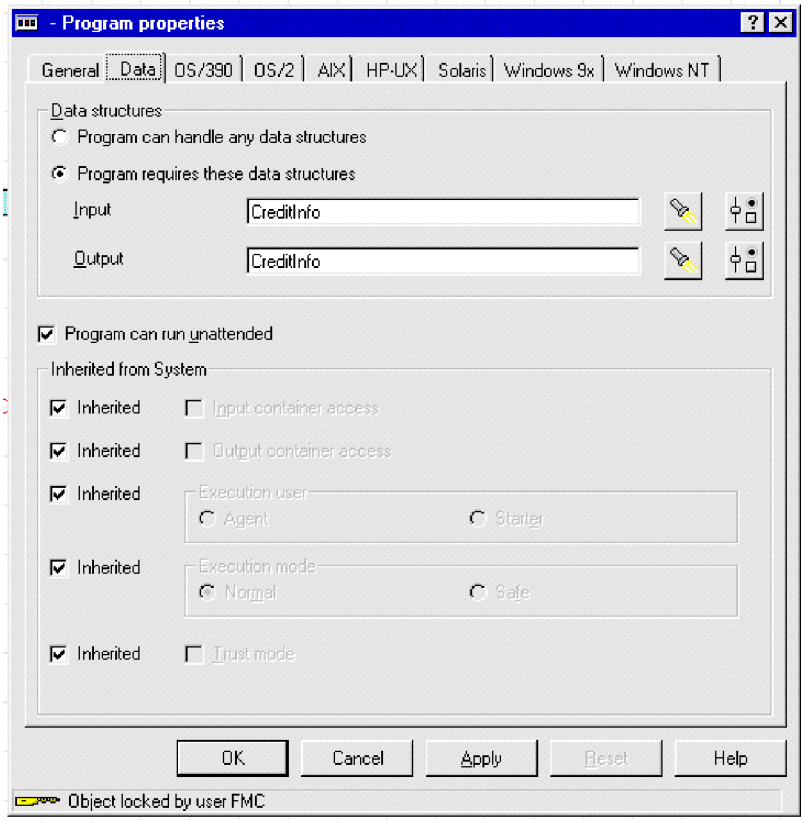

Figure 19. WebSphere MQ Workflow Buildtime: Specifying data structures

9. Click the Windows tab and enter an existing program to execute.

   Note: Although the program you specify is not executed, WebSphere MQ Workflow requires that it be defined.

Two command line parameters must be specified at workflow design time that indicate which verb and collaboration to use when posting the data structure to ICS. The connector requires that these parameters follow a name-value format and that multiple name-value pairs be delimited by semi-colons. Currently, two values can be specified: `verb` and *`collab`*. For example, to specify that the workflow data structure be issued to the connector and then processed with an `Update` verb in collaboration `CreditInfoSync`, the program parameters must equal `verb=Update; collab=CreditInfoSync`. If a collaboration name is not specified (a `verb=Update` program parameter), the data structure is posted to all subscribing collaborations.

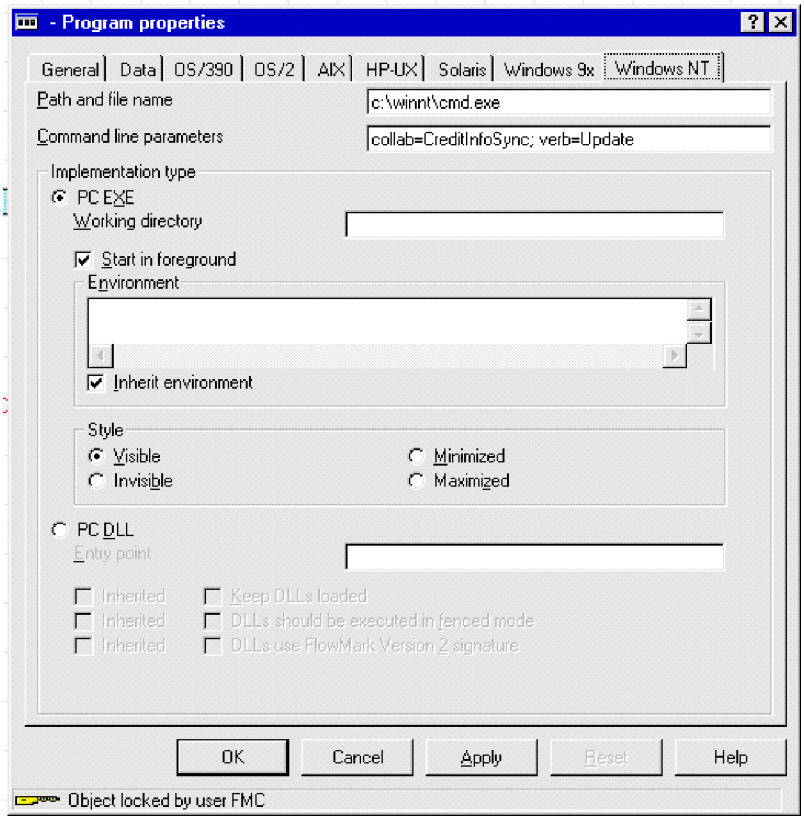

Figure 20. WebSphere MQ Workflow Buildtime: Specifying command-line parameters

10. To make a program node issue requests to the WebSphere MQ Workflow connector, create a new program node and specify the name of the program (as defined in step 7).

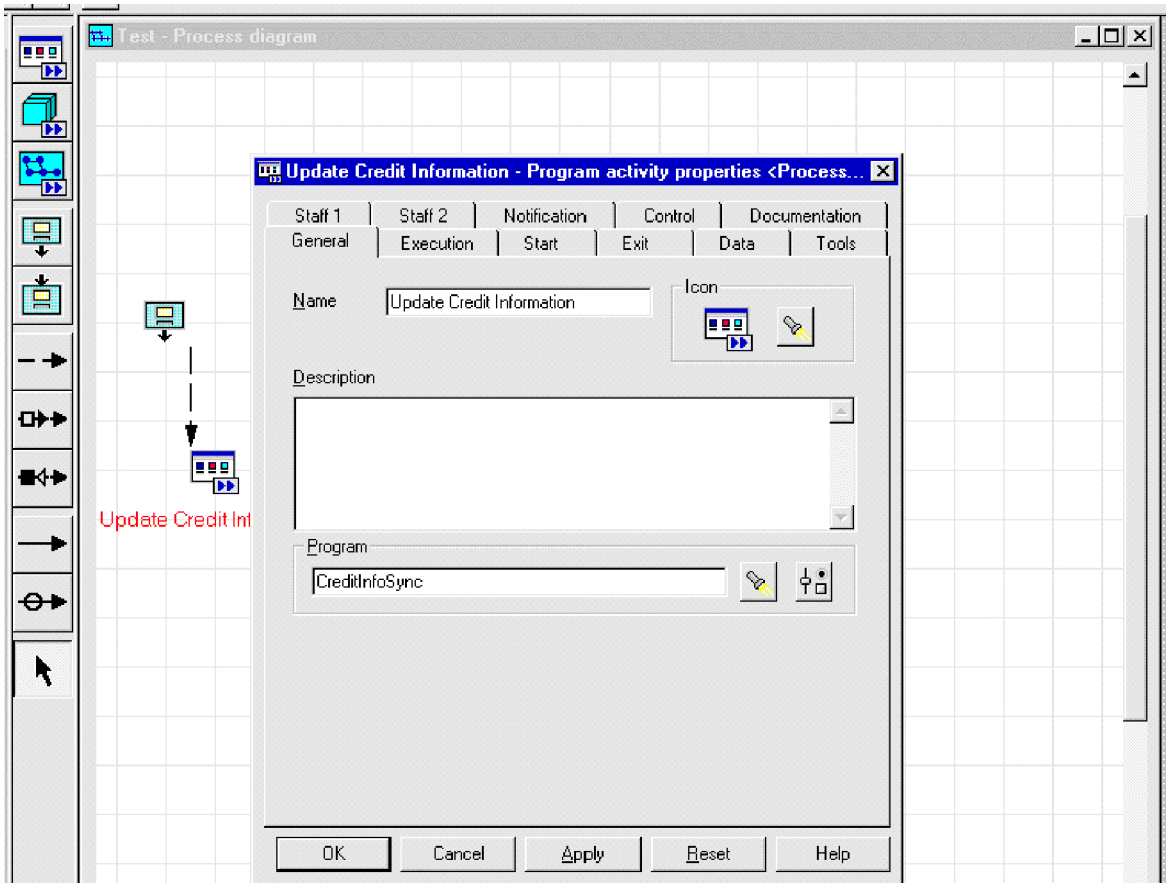

Figure 21. WebSphere MQ Workflow Buildtime: Creating the new program node

11. Define the program execution server (`CWLDSVR.FMCSYS.FMCGRP`) and select either synchronous or asynchronous for the type of request.

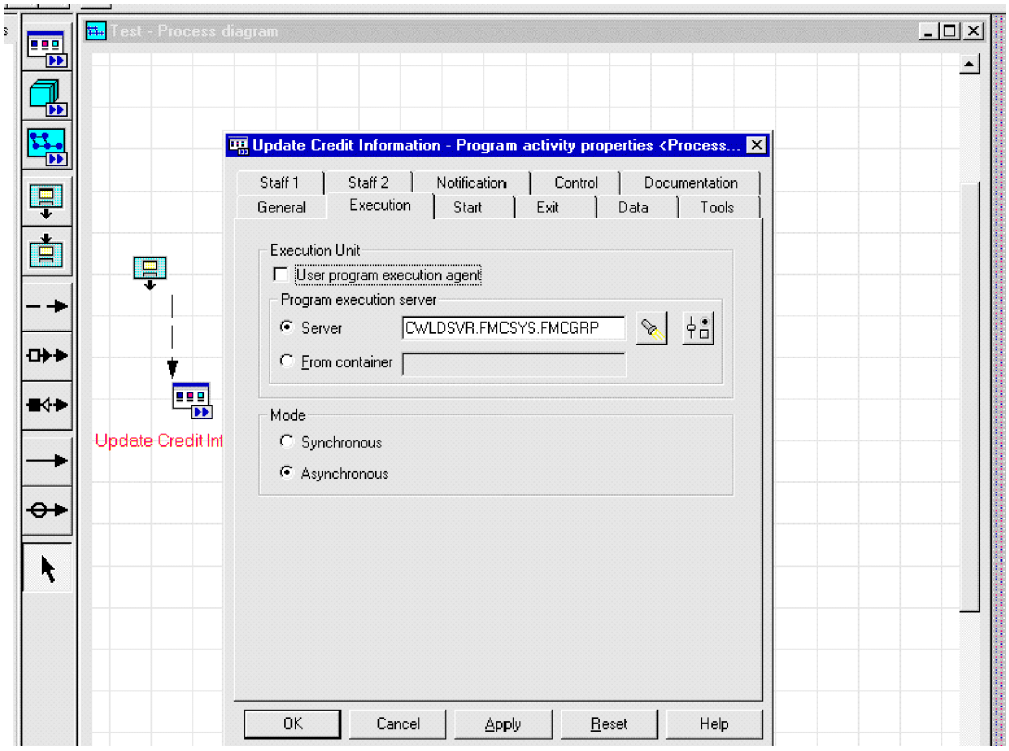

Figure 22. WebSphere MQ Workflow Buildtime: Defining the program server

When this node is reached in WebSphere MQ Workflow, a message containing the workflow data structure is issued to the connector. The connector processes the business content and returns any changes or errors with the content.

Chapter 4. Developing business objects

After installing and configuring the connector for WebSphere MQ Workflow, you must create business objects. You can do this by:

- Modifying sample business objects that are shipped with this release
- Extracting business object definition files from WebSphere MQ Workflow export files

This chapter provides information you can use as a guide to implement new business objects. It also shows you sample business objects you can modify. To extract business objects definition files from WebSphere MQ Workflow, see Chapter 5, "Using the FDLBORGEN utility to create business object definitions," on page 69

Connector business object structure

The connector is a metadata-driven connector. In business objects, metadata is data about the application, which is stored in a business object definition and which helps the connector interact with an application. A metadata-driven connector handles each business object that it supports based on metadata encoded in the business object definition rather than on instructions hard-coded in the connector.

Business object metadata includes the structure of a business object, the settings of its attribute properties, and the content of its application-specific information. Because the connector is metadata-driven, it can handle new or modified business objects without requiring modifications to the connector code. However, the connector's configured data handler makes assumptions about the structure of its business objects. Therefore, when you create or modify a business object for WebSphere MQ Workflow, your modifications must conform to the rules the connector is designed to follow, or the connector cannot process the new or modified business objects correctly.

There are requirements regarding the structure of the business objects other than those imposed by the XML data handler. For more on this, see "Meta-object configuration" on page 32. The business objects that the connector processes can have any name allowed by InterChange Server.

The connector retrieves messages from a queue and attempts to populate a business object (defined by the top-level business object and metadata) with the message contents. Strictly speaking, the connector neither controls nor influences business object structure. Those are functions of meta-object definitions as well as the connector's data handler requirements. The connector's main role when retrieving and passing business objects is to monitor the message-to-business-object (and vice versa) process for errors.

Sample business object definitions

This section provides sample business object definitions. For specific information on business object attributes such as Cardinality, IsKey, and so on, see the *Connector Development Guide for Java*.

```
[BusinessObjectDefinition]
Name = MQWF_SampleItem
Version = 1.0.0
AppSpecificInfo = cw_mo_wfcontainer=ContainerInfo

[Attribute]
Name = Input_Item
Type = MQWF_Structure_SampleItem
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = SampleItem;type=pcdata;
IsRequiredServerBound = false
[End]
[Attribute]
Name = ContainerInfo
Type = MO_MQWorkflow_ContainerInfo
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Output_Item
Type = MQWF_Structure_SampleItem
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = SampleItem;type=pcdata;
IsRequiredServerBound = false
[End]
[Attribute]
Name = ObjectEventId
Type = String
MaxLength = 255
IsKey = true
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 = MQWF_Structure_SampleItem
Version = 1.0.0
AppSpecificInfo = SampleItem

[Attribute]
Name = Name
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = Name;type=pcdata;
DefaultValue =
[End]

[Attribute]
Name = Price
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = Price;type=pcdata;
DefaultValue =
[End]

[Attribute]
Name = Stock
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = Stock;type=pcdata;
DefaultValue =
[End]

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

[Verb]
Name = Create
[End]

[Verb]
```

```
Name = Retrieve
[End]

[Verb]
Name = Update
[End]

[Verb]
Name = Delete
[End]
[End]

[BusinessObjectDefinition]
Name = MO_MQWorkflow_ProcessInfo
Version = 1.0.0

[Attribute]
Name = Role
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Organization
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = ProcessAdministrator
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Duration
Type = String
Cardinality = 1
MaxLength = 255
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]

[BusinessObjectDefinition]
Name = MO_MQWorkflow_ActivityInfo
Version = 1.0.0

[Attribute]
Name = Priority
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = MembersOfRoles
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = CoordinatorOfRole
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Organization
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = OrganizationType
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
```

```
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = LowerLevel
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = UpperLevel
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = People
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = PersonToNotify
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Duration
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = Duration2
Type = String
Cardinality = 1
MaxLength = 255
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]

[BusinessObjectDefinition]
Name = MO_MQWorkflow_ContainerInfo
Version = 1.0.0

[Attribute]
Name = PROCESS_INFO
Type = MO_MQWorkflow_ProcessInfo
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = ACTIVITY_INFO
Type = MO_MQWorkflow_ActivityInfo
ContainedObjectVersion = 1.0.0
Relationship = Containment
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = ACTIVITY
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
IsRequiredServerBound = false
[End]
[Attribute]
Name = PROCESS
Type = String
Cardinality = 1
MaxLength = 255
IsKey = true
```

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

Error handling

All error messages generated by the connector are stored in a message file. (The name of the file is determined by the `LogFileName` standard connector configuration property.) Each error has an error number followed by the error message:

```
Message number
Message text
```

The connector handles specific errors as described in the following sections.

Application timeout

The error message BON_APPRESPONSETIMEOUT is returned when:

- The connector cannot establish a connection to the WebSphere MQ Workflow queue manager during message retrieval.
- The connector successfully converts a business object to a message but cannot deliver it to the outgoing queue due to connection loss.
- The connector issues a message but times out waiting for a response for a business object with conversion property `TimeoutFatal` equal to `true`.

- The connector times out while communicating with WebSphere MQ Workflow using the synchronous API.

Unsubscribed message

The connector delivers a message to the queue specified by the `UnsubscribedQueue` property if:

- The connector retrieves a message that does not have a format of either MQSTR or FMXML.
- The connector retrieves a `WfMessage`, but the message name is not of type `ActivityImplInvoke` or `General Error`
- The connector cannot find the top-level business object for the data structure when processing a subscription delivery.

Note: If the `UnsubscribedQueue` is not defined, unsubscribed messages are discarded.

XML structure errors

The connector delivers a message to the queue specified by the `ErrorQueue` property if:

- The XML document is not well formed
- The `WfMessage` structure is not complete.

Data handler conversion

The connector delivers a message to the queue specified by the `ErrorQueue` property if:

- The data handler cannot convert a business object to XML or vice-versa.

Errors and the WfMessage document

In addition to responding to standard errors reported by the connector, the connector also responds to errors issued by WebSphere MQ Workflow itself. If an error occurs while the connector is synchronously processing a message issued by the connector, WebSphere MQ Workflow returns a WfMessage containing an Exception element. The text of this element specifies an error message that the connector returns to InterChange Server for failed request operations to the WebSphere MQ Workflow application.

If the connector fails to synchronously process a WfMessage issued by WebSphere MQ Workflow (for any of the aforementioned reasons), the connector attempts to send WebSphere MQ Workflow a response WfMessage containing an Exception element. The connector populates this element with a verbose explanation of the event or events that caused the failure. WebSphere MQ Workflow can use this message to update the state of a failed workflow so that appropriate user intervention can be taken.

Once the connector issues a response to a request that was originally issued by the WebSphere MQ Workflow server, the connector does not wait for an acknowledgement. The connector has no means of receiving an acknowledgement because WebSphere MQ Workflow reports errors only in such cases. To circumvent this problem, the connector specifies in its response message that any error message resulting from the response should be issued to the input queue of the connector. In this fashion, if an error occurs, the connector is eventually notified during `pollForEvents`. The error is logged, but no further action is taken. Such

errors are assumed to be the consequence of response business objects that produce incomplete or incorrect data structures as determined by WebSphere MQ Workflow.

Tracing

Tracing is an optional debugging feature you can turn on to closely follow connector behavior. Trace messages, by default, are written to STDOUT. See the connector configuration properties in Chapter 3, "Modifying the WebSphere MQ Workflow application," on page 51for more on configuring trace messages. For more information on tracing, including how to enable and set it, see the *Connector Development Guide*.

The following level options are available for connector trace messages.

Level 0	The connector lists only its version.
Level 1	The connector logs each time a message is retrieved during polling or posted during request processing.
Level 2	The connector logs each time a business object is posted to InterChange Server, either from `gotApplEvent()` or `executeCollaboration()`.
Level 3	The connector provides increased details of the messages being parsed and the logic executed in this process.
Level 4	The connector traces each method it enters or exits.
Level 5	The connector logs its initialization, dumps all business objects processed and lists all XML documents being read or written.

Chapter 5. Using the FDLBORGEN utility to create business object definitions

This chapter describes FDLBORGEN, a stand-alone command-line utility that automates the extraction of business object definition files from WebSphere MQ Workflow export files. The resulting business object definition files are compatible with the XML data handler.

FDL, the acronym for the WebSphere MQ Workflow Definition Language, is the suffix of WebSphere MQ Workflow export files that FDLBORGEN uses as input.

About the FDLBORGEN utility

The FDLBORGEN utility:

- Creates business object definitions based on an FDL structure definition and writes them to a business object definition file. This file can be loaded into InterChange Server repository using the `repos_copy` utility.
- Builds business object definitions that conform to the requirements of the XML data handler. These business object definitions do not need additional editing.
- Adds the required `ObjectEventId` attribute to all business object definitions. It also adds the repository version number to the top of the business object file if you specify this.

FDLBORGEN consists of two files that are installed as part of the `connectors\MQWorkflow\utilities\fdlborgen` directory (for a complete installed product directories, see "Installed file structure" on page 19):

- An executable file that invokes the FDLBORGEN utility:
  - In Windows environments, this executable file is `fdlborgen.bat`.
  - In UNIX environments, this executable file is `fdlborgen`.
- A Java archive file, `fdlborgen.jar`, which contains all the Java classes necessary for FDLBORGEN's operation.

Before using the FDLBORGEN utility

Before you can use the FDLBORGEN utility, you must create one or more input files. You do this by generating and exporting a file of type `.fdl` from WebSphere MQ Workflow.

1. Open WebSphere MQ Workflow Buildtime.
2. Select Export from the Buildtime menu.
3. In the dialog box, choose a filename (for example, `MyExport.fdl`) and click OK.

FDLBORGEN syntax

To run the FDLBORGEN command-line utility:

1. Open a DOS command prompt window (Windows) or a shell window (UNIX).
2. Enter the command for FDLBORGEN.

The format for FDLBORGEN command is defined as:

```
fdlborgen -i[input-file] -o[output-file] -p[prefix]
 {-n[object-name]} {-r[repos_version]} {-v[verblist]} {-V}
```

where:

`-i[input-file]`	Specifies the name of the `.fdl` file and its path, if the `fdl` file is not located in the current directory.
`-o[output-file]`	Specifies the name and location where the generated business object definition will be stored.
`-n[object-name]`	Specifies the name of the top-level data structure in the `fdl` file that is to be converted to a business object definition.
`-p[prefix]`	Specifies the prefix that is inserted before the name of the generated business object definition. This is useful when differentiating between the `fdl` object name and the business object name. The prefix option is required.

`-r[repos_version]`	Adds the ReposCopy header to the beginning of the generated business object definition file. For example, `-r1.0.2` adds the following to the beginning of the file:
	`[ReposCopy]`
	`Version = 1.0.2`
	`[End]`
	
	If you are overwriting an existing version of a business object definition file, use this parameter to preserve the version information.

`-v[verblist]`	Specifies the list of verbs to be included in each business object. The verbs Create, Retrieve, Update, and Delete are supported. Separate each verb with a comma and do not add spaces. If you do not specify this parameter, the standard Create, Retrieve, Update, and Retrieve verbs are added.
`-V`	Switches the program into verbose mode and prints out all entries, attributes, elements, and comments encountered.
`-[mapping-file]`	Specifies the name of a mapping file (and its path if the file is not located in the current directory). See ″Notes on Conversion″ below about the mapping file.

Note: All errors encountered from FDLBORGEN are sent to `stderr`.

FDLBORGEN example

To convert the WebSphere MQ Workflow data structure `MyCustomer` to business object `Wf_MyCustomer`, you must first generate an input file in WebSphere MQ Workflow Buildtime. Using `MyExport.fdl` in the current directory, you can execute FDLBORGEN as follows:

```
fdlborgen –iMyExport.FDL –oMyBO.txt –nMyCustomer –pWf_ -vCreate,Update –r2.0.0
```

This generates business object `Wf_MyCustomer` based upon data structure `MyCustomer` in FDL file `MyExport.FDL`. Any required child data structures are also included in this definition file.

Notes on conversion

Any member of a data structure of type `STRING`, `LONG`, or `FLOAT` is mapped as an attribute of the same name and of type `STRING`.

Any object member in a data structure is converted to a child object of the same name but of type <boprefix><*data structure name*>.

The business object names and the attribute names can have up to 80 alphanumeric characters and underscores. Abide by this convention when naming the top-level data structure or the member name in the fdl. Otherwise, prepare a mapping table and specify the mapping table file using the -m command option.

The mapping table file should be in the following CSV format:

- `<name1 in FDL>,<mapped name1>`
- `<name2 in FDL>,<mapped name2>`

When you specify the -m option and the mapping file name and path (as needed), the

`<name1 in FDL>`

is replaced with

`<mapped name1>`

in the generated business object definition file.

Chapter 6. Troubleshooting

This chapter describes problems that you may encounter when starting up or running the connector for WebSphere MQ Workflow.

Startup problems

Problem	Potential solution / explanation
The connector shuts down unexpectedly during initialization and the following message is reported: `java.lang.NoSuchMethodError at com.ibm.workflow.api.Agent$OsaLocator. locateController(Agent.java:219) at com.ibm.workflow.api.Agent$OrbLocator. locate(Agent.java:173) at com.ibm.workflow.api.Agent. setName(Agent.java:401).`	This error may indicate that the default IBM Java ORB libraries are incompatible with those required by the WebSphere MQ Workflow API for communication to a remote Workflow server via an IBM Java ORB agent. Per IBM requirements, these libraries need to be bootstrapped so that they take precedence over the default IBM Java ORB libraries. To do this, open `start_connector.bat` or `start_connector.sh`), scroll down to the section that begins STEP 3 and ensure that you have specified the correct IBM Java ORB libraries as required by IBM. **Note:** The WebSphere MQ Workflow API may require different IBM Java ORB libraries than are shipped. See *IBM WebSphere MQ Workflow: Programming Guide* for more information.
The connector terminates while processing a message and reports an error similar to the following:`Exception thrown: com.crossworlds.connectors.mqworkflow.exceptions. FatalProcessingException: [Type: Fatal Error ] [MsgID: 40007] [Mesg: Failed to read message content. An IO error occurred: java.io.UnsupportedEncodingException Cp437.] ]`	The JVM version you are using does not have the libraries necessary to support the message character set. The easiest solution to this problem is to download a more recent version of the JDK from Sun Microsystems and run the connector in this new JVM. Open the `start_connector.bat` (or `start_connector.sh`) file and replace all instances of `%CROSSWORLDS%\bin\java` with the path of your new JVM.
The connector shuts down unexpectedly during initialization and the following exception is reported: `java.lang.UnsatisfiedLinkError: no mqjbnd01 in shared library path`	Connector cannot find a required runtime library (`mqjbnd01.dll` [Windows] or `libmqjbnd01.so` [UNIX]) from the IBM WebSphere MQ Java client libraries. Ensure that your path includes the library folder.
The connector shuts down unexpectedly during initialization, and the following message is reported:`Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/workflow/api/FmcException`	Verify that you specified the correct path of your MQWorkflow client libraries in `start_WebSphereMQWorkflow.bat` (Windows) or `start_WebSphereMQWorkflow.sh` (UNIX). See “Startup file configuration” on page 46 for further instructions.

Problem	Potential solution / explanation
The connector shuts down unexpectedly during initialization and the following exception is reported: `java.lang.UnsatisfiedLinkError: no fmcojprf (libfmcojpfr.a or .so) in java.library.path`	The connector cannot find a required runtime library. Search for the library specified (for example, `libfmcojprf.a`) on your system and ensure that the parent directory for this file is included in your path. If the library cannot be found, ensure that you've installed all necessary prerequisite software (WebSphere MQ Workflow application and WebSphere MQ client libraries). Once the library is located and added to your path, you may need to modify the `start_connector` script and then add this path to the `java.library.path` option that is passed in the command line to start the adapter. To do this, scroll to the bottom of the start script and locate the command that starts the adapter (this may begin with *ProductDir*`\bin\java`). The command line specifies options. Locate the option that begins `_Djava.library.path=`. Append the parent directory for the library in question to the list of other directories specified for the `_Djava.library.path` option. For example, if `c:program\files\webspheremq workflow\bin` contained `libfmcojprf.a`.,after modification, your command line might include `_Djava.library.path=`*ProductDir*`\bin; %CONNDIR%;c:\program files\webspheremqworkflow\bin`.

Event processing

Problem	Potential solution / explanation
The connector sends all messages with a user ID of "fmc" regardless of what is specified in the connector configuration.	Examine the properties of the WebSphere MQ server channel specified for the connector. Verify that no value is specified for its MCA User ID property.

Appendix A. Standard configuration properties for connectors

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

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

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

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

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

New and deleted properties

These standard properties have been added in this release.

New properties

- XMLNameSpaceFormat

Deleted properties

- RestartCount
- RHF2MessageDomain

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

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

Using Connector Configurator

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

Note: Connector Configurator and System Manager run only on the Windows system. If you are running the connector on a UNIX system, you must have

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

Setting and updating property values

The default length of a property field is 255 characters.

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

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

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

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

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 the Property Summary table below.

Summary of standard properties

Table 13 on page 77 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, as standard property dependencies are based on `RepositoryDirectory`.

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

Table 13. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdminInQueue	Valid JMS queue name	*CONNECTORNAME* /ADMININQUEUE	Component restart	Delivery Transport is JMS
AdminOutQueue	Valid JMS queue name	*CONNECTORNAME*/ADMINOUTQUEUE	Component restart	Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	Delivery Transport is MQ or IDL: Repository directory is <REMOTE>
AgentTraceLevel	0-5	0	Dynamic	
ApplicationName	Application name	Value specified for the connector application name	Component restart	
BrokerType	ICS, WMQI, WAS			
CharacterEncoding	ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437 **Note:** This is a subset of supported values.	ascii7	Component restart	
ConcurrentEventTriggeredFlows	1 to 32,767	1	Component restart	Repository directory is <REMOTE>
ContainerManagedEvents	No value or JMS	No value	Component restart	Delivery Transport is JMS
ControllerStoreAndForwardMode	true or false	True	Dynamic	Repository directory is <REMOTE>
ControllerTraceLevel	0-5	0	Dynamic	Repository directory is <REMOTE>
DeliveryQueue		*CONNECTORNAME*/DELIVERYQUEUE	Component restart	JMS transport only
DeliveryTransport	MQ, IDL, or JMS	JMS	Component restart	If Repository directory is local, then value is JMS only
DuplicateEventElimination	True or False	False	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		*CONNECTORNAME*/FAULTQUEUE	Component restart	JMS transport only

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
jms.FactoryClassName	`CxCommon.Messaging.jms .IBMMQSeriesFactory` or `CxCommon.Messaging .jms.SonicMQFactory` or any Java class name	`CxCommon.Messaging. jms.IBMMQSeriesFactory`	Component restart	JMS transport only
jms.MessageBrokerName	If FactoryClassName is `IBM`, use `crossworlds.queue. manager`. If FactoryClassName is `Sonic`, use `localhost:2506`.	`crossworlds.queue.manager`	Component restart	JMS transport only
jms.NumConcurrentRequests	Positive integer	`10`	Component restart	JMS transport only
jms.Password	Any valid password		Component restart	JMS transport only
jms.UserName	Any valid name		Component restart	JMS transport only
JvmMaxHeapSize	Heap size in megabytes	`128m`	Component restart	Repository directory is <REMOTE>
JvmMaxNativeStackSize	Size of stack in kilobytes	`128k`	Component restart	Repository directory is <REMOTE>
JvmMinHeapSize	Heap size in megabytes	`1m`	Component restart	Repository directory is <REMOTE>
ListenerConcurrency	`1- 100`	`1`	Component restart	Delivery Transport must be `MQ`
Locale	`en_US`, `ja_JP`, `ko_KR`, `zh_CN`, `zh_TW`, `fr_FR`, `de_DE`, `it_IT`, `es_ES`, `pt_BR` **Note:** This is a subset of the supported locales.	`en_US`	Component restart	
LogAtInterchangeEnd	`True` or `False`	`False`	Component restart	Repository Directory must be <REMOTE>
MaxEventCapacity	`1-2147483647`	`2147483647`	Dynamic	Repository Directory must be <REMOTE>
MessageFileName	Path or filename	`InterchangeSystem.txt`	Component restart	
MonitorQueue	Any valid queue name	*`CONNECTORNAME`*`/MONITORQUEUE`	Component restart	JMS transport only: DuplicateEvent Elimination must be True
OADAutoRestartAgent	`True` or `False`	`False`	Dynamic	Repository Directory must be <REMOTE>

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
OADMaxNumRetry	A positive number	1000	Dynamic	Repository Directory must be <REMOTE>
OADRetryTimeInterval	A positive number in minutes	10	Dynamic	Repository Directory must be <REMOTE>
PollEndTime	HH:MM	HH:MM	Component restart	
PollFrequency	A positive integer in milliseconds no (to disable polling) key (to poll only when the letter p is entered in the connector's Command Prompt window)	10000	Dynamic	
PollQuantity	1-500	1	Agent restart	JMS transport only: Container Managed Events is specified
PollStartTime	HH:MM(HH is 0-23, MM is 0-59)	HH:MM	Component restart	
RepositoryDirectory	Location of metadata repository		Agent restart	For ICS: set to <REMOTE> For WebSphere MQ message brokers and WAS: set to C:\crossworlds\ repository
RequestQueue	Valid JMS queue name	*CONNECTORNAME*/REQUESTQUEUE	Component restart	Delivery Transport is JMS
ResponseQueue	Valid JMS queue name	*CONNECTORNAME*/RESPONSEQUEUE	Component restart	Delivery Transport is JMS: required only if Repository directory is <REMOTE>
RestartRetryCount	0-99	3	Dynamic	
RestartRetryInterval	A sensible positive value in minutes: 1 - 2147483547	1	Dynamic	
SourceQueue	Valid WebSphere MQ name	*CONNECTORNAME*/SOURCEQUEUE	Agent restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	Component restart	Delivery Transport is JMS

Table 13. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
SynchronousRequestTimeout	0 - any number (millisecs)	0	Component restart	Delivery Transport is JMS
SynchronousResponseQueue		*CONNECTORNAME*/SYNCHRONOUSRESPONSEQUEUE	Component restart	Delivery Transport is JMS
WireFormat	CwXML, CwBO	CwXML	Agent restart	CwXML if Repository Directory is not <REMOTE>: CwBO if Repository Directory is <REMOTE>
WsifSynchronousRequest Timeout	0 - any number (millisecs)	0	Component restart	WAS only
XMLNameSpaceFormat	short, long	short	Agent restart	WebSphere MQ message brokers and WAS only

Standard configuration properties

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

AdminInQueue

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

The default value is CONNECTORNAME/ADMININQUEUE.

AdminOutQueue

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

The default value is CONNECTORNAME/ADMINOUTQUEUE.

AgentConnections

Applicable only if RepositoryDirectory is <REMOTE>.

The AgentConnections property controls the number of ORB connections opened by orb.init[].

By default, the value of this property is set to 1. There is no need to change this default.

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

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

ConcurrentEventTriggeredFlows

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

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

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

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

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

ContainerManagedEvents

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

The default value is No value.

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

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

You must also configure a data handler with the MimeType, DHClass, and DataHandlerConfigMOName (optional) properties. To set those values, use the **Data Handler** tab in Connector Configurator. The fields for the values under the Data Handler tab will be displayed only if you have set ContainerManagedEvents to JMS.

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

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

ControllerStoreAndForwardMode

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

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

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

The default is true.

ControllerTraceLevel

Applicable only if RepositoryDirectory is <REMOTE>.

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

DeliveryQueue

Applicable only if DeliveryTransport is JMS.

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

The default value is CONNECTORNAME/DELIVERYQUEUE.

DeliveryTransport

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

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

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

WebSphere MQ and IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

This property can also be set to `false`.

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

FaultQueue

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

The default value is `CONNECTORNAME/FAULTQUEUE`.

JvmMaxHeapSize

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

The default value is 128m.

JvmMaxNativeStackSize

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

The default value is 128k.

JvmMinHeapSize

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

The default value is 1m.

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

The default is `crossworlds.queue.manager`.

jms.NumConcurrentRequests

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

The default value is 10.

jms.Password

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

There is no default.

jms.UserName

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

There is no default.

ListenerConcurrency

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

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

Locale

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

A locale name has the following format:

ll_TT.codeset

where:

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

By default, only a subset of supported locales appears in the drop list. To add other supported values to the drop list, you must manually modify the

\Data\Std\stdConnProps.xml file in the product directory. For more information, see the appendix on Connector Configurator.

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

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

LogAtInterchangeEnd

Applicable only if RespositoryDirectory is <REMOTE>.

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

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

MaxEventCapacity

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

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

MessageFileName

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

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

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

MonitorQueue

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

The default value is CONNECTORNAME/MONITORQUEUE

OADAutoRestartAgent

Valid only when the RepositoryDirectory is <REMOTE>.

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

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

The default value is `false`.

OADMaxNumRetry

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

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

The default value is `1000`.

OADRetryTimeInterval

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

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

The default is `10`.

PollEndTime

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

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

PollFrequency

The amount of time between polling actions. Set `PollFrequency` to one of the following values:

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

The default is `10000`.

Important: Some connectors have restrictions on the use of this property. To determine whether a specific connector does, see the installing and configuring chapter of its adapter guide.

PollQuantity

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

PollStartTime

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

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

RequestQueue

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

The default value is `CONNECTOR/REQUESTQUEUE`.

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

The default is `3`.

RestartRetryInterval

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

The default is 1.

SourceQueue

Applicable only if DeliveryTransport is JMS and ContainerManagedEvents is specified.

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see "ContainerManagedEvents" on page 81.

The default value is CONNECTOR/SOURCEQUEUE.

SynchronousRequestQueue

Applicable only if DeliveryTransport is JMS.

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

The default is CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE

SynchronousResponseQueue

Applicable only if DeliveryTransport is JMS.

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

The default is CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE

SynchronousRequestTimeout

Applicable only if DeliveryTransport is JMS.

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

The default value is 0.

WireFormat

Message format on the transport.

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

WsifSynchronousRequest Timeout

WAS integration broker only.

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

The default value is 0.

XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

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

The default value is `short`.

Appendix B. Connector Configurator

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

You use Connector Configurator to:

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

Note:

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

The topics covered in this appendix are:

Overview of Connector Configurator

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

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

You use Connector Configurator to:

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

The mode in which you run Connector Configurator, and the configuration file type you use, may differ according to which integration broker you are running. For example, if WMQI is your broker, you run Connector Configurator directly, and not from within System Manager (see "Running Configurator in stand-alone mode" on page 92).

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 93 to set up a new one.

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

Starting Connector Configurator

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

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

Running Configurator in stand-alone mode

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

To do so:

- From **Start>Programs**, click **IBM WebSphere InterChange Server>IBM WebSphere Business Integration Toolset>Development>Connector Configurator**.
- Select **File>New>Configuration File**.
- 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 97.)

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:

1. 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.
2. 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 file as the template.

- To create a new template, see "Creating a new template" on page 93.
- To use an existing file, simply modify an existing template and save it under the new name.

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:

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

     Enter a unique name that identifies the connector, or type of connector, for which this template will be used. You will see this name again when you open the dialog box for creating a new configuration file from a template.
   - **Old Template**, and **Select the Existing Template to Modify**

     The names of all currently available templates are displayed in the Template Name display.

- 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 will appear in the **Template Preview** display. You can use an existing template whose property definitions are similar to those required by your connector as a starting point for your template.

3. Select a template from the **Template Name** display, enter that template name in the **Find Name** field (or highlight your selection in **Template Name**), and click **Next**.

If you do not see any template that displays the connector-specific properties used by your connector, you will need to create one.

Specifying general characteristics

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

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

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

Specifying values

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

1. Click the **Value** tab. The display panel for Value replaces the display panel for General.
2. Select the name of the property in the **Edit properties** display.
3. In the fields for **Max Length** and **Max Multiple Values**, make any changes. The changes will not be accepted unless you also open the **Property Value** dialog box for the property, described in the next step.
4. Right-click the box in the top left-hand corner of the value table and click **Add**. A **Property Value** dialog box appears. Depending on the property type, the dialog box allows you to enter either a value, or both a value and range. Enter the appropriate value or range, and click **OK**.
5. The **Value** panel refreshes to display any changes you made in **Max Length** and **Max Multiple Values**. It 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 **Dependences - Connector-Specific Property Template** dialog box appears.

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

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

Creating a new configuration file

When you create a new configuration file, your first step is 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 Connector Configurator home menu, click **File>New>Connector Configuration**. The **New Connector** dialog box appears.
- In the I**ntegration 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.

You can also do this:

- In the System Manager window, right-click on the **Connectors** folder and select **Create New Connector**. Connector Configurator opens and displays the **New Connector** dialog box.

Creating a configuration file from a connector-specific template

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

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

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

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

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

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

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

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

Using an existing file

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

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

- A previous configuration file for the connector.
  Such a file typically has the extension *.cfg.

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

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

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

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

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

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

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

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

Completing a configuration file

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

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

1. Under the **Standard Properties** tab, select the value field for the `BrokerType` property. In the drop-down menu, select the value `ICS`, `WMQI`, or `WAS`.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in “Specifying supported business object definitions” on page 100..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)

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

Standard properties differ from connector-specific properties as follows:

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

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

- A field with a grey background indicates a standard property. You can change the value but cannot change the name or remove the property.
- A field with a white background indicates an application-specific property. These properties vary according to the specific needs of the application or connector. You can change the value and delete these properties.
- Value fields are configurable.
- The **Update Method** field is informational and not configurable. This field specifies the action required to activate a property whose value has changed.

Setting standard connector properties

To change the value of a standard property:

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

Setting application-specific configuration properties

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

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

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

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

Encryption for connector properties

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

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

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

Update method

Refer to the descriptions of update methods found in the *Standard configuration properties for connectors* appendix, under "Setting and updating property values" on page 76.

Specifying supported business object definitions

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

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

If ICS is your broker

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

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

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

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

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

If WAS is your broker

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

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

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

Associated maps (ICS only)

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

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

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

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

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

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

- **Business Object Name**

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

- **Associated Maps**

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

- **Explicit**

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

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

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

  To explicitly bind a map:

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

Resources (ICS)

The **Resource** tab allows you to set a value that determines whether and to what extent the connector agent will handle multiple processes concurrently, using connector agent parallelism.
Not all connectors support this feature. If you are running a connector agent that was designed in Java to be multi-threaded, you are advised not to use this feature, since it is usually more efficient to use multiple threads than multiple processes.

Messaging (ICS)

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

Setting trace/log file values

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

To change the logging and tracing values:

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

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

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

Data handlers

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

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

Saving your configuration file

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

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

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

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

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

Changing a configuration file

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

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

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

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

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

Appendix C. Tutorial

This appendix describes how to install, configure, and work with the samples that are shipped with the adapter. This tutorial is designed to work with an InterChange Server (ICS) integration broker. See the Preface of this document for a guide to notational conventions.

Prerequisites

Before using this tutorial you must install and be familiar with the IBM WebSphere Business Integration product. Complete the following tasks before installing the samples:

1. Install the adapter for WebSphere MQ Workflow. The samples in this tutorial are designed for WebSphere MQ Workflow version 3.2.2, 3.3.2 and 3.4.x; support for other versions may vary.
2. Install the Port connector. (There is no actual agent associated with the Port connector, but the connector definition must exist in your repository.)
3. If you do not have a WebSphere MQ Workflow Adapter definition and Port Connector definition in the Repository, then use Connector Configurator to load the them:
   a. Select the File" Open From File menu item.
   b. Load the repository file `Sample_MQWorkflow_Order_Connectors.in` located in the `\connectors\WebSphereMQWorkflow\samples` folder.
   c. Confirm that the MQWorkflowConnector and PortConnector definitions have been loaded.
4. Install the IBM WebSphere MQ client libraries for Java.

Pre-install checklist

Before installing the samples, gather the following information:

- The name of your IBM WebSphere InterChange Server (default: `LocalHost`)
  ICS name = _____
- Your WebSphere MQ Workflow queue manager (default: `FMCQM`)
  queue manager name = _____
- The CCSID of the queue manager (default: `819`)
  queue manager CCSID = _____
- The port established for the queue manager listener (default: `5010`)
  queue manager port = _____
- The host for the queue manager (default: `LocalHost`)
  queue manager host = _____
- The server connection channel used for the queue manager (default: `FMCQM.CL.TCP`)
  queue manager channel = _____
- Your WebSphere MQ Workflow system name (default: `FMCSYS`)
  WebSphere MQ Workflow system name = _____
- Your WebSphere MQ Workflow system group (default: `FMCGRP`)
  WebSphere MQ Workflow system group = _____
- Your WebSphere MQ Workflow account user name (default: `ADMIN`)
  WebSphere MQ Workflow account user name = _____

Note: This account must exist in WebSphere MQ Workflow and as part of group MQM in your system accounts. Failure to ensure this may prevent the adapter from issuing a message via WebSphere MQ or executing a WebSphere MQ Workflow process in WebSphere MQ Workflow.

- Your WebSphere MQ Workflow configuration name (default: FMC)
  WebSphere MQ Workflow configuration name = _____

Setting up your environment

This section describes how to prepare your environment to work with the samples. In what follows, *sample_folder* refers to the folder in which the samples reside, and*WBI_folder* refers to the folder containing your current IBM WebSphere Business Integration installation.

1. **Create the queues** This tutorial requires that six local queues be defined in WebSphere MQ Workflow queue manager. You create these either via the WebSphere MQ Explorer application or by typing RUNMQSC FMCQM at the command line and issuing the following commands:
   - DEFINE QL('MQWFCONN.ERROR')
   - DEFINE QL('MQWFCONN.ARCHIVE')
   - DEFINE QL('MQWFCONN.IN_PROGRESS')
   - DEFINE QL('MQWFCONN.REPLYTO')
   - DEFINE QL('MQWFCONN.UNSUBSCRIBED')
   - DEFINE QL('CWLDINPUTQ')
2. **Create business object definitions—optional** Business object definition files for the WebSphere MQ Workflow data structures already exist, but performing this step demonstrates the FDLBORGEN utility, which converts data structures to business objects. To use this utility:
   a. From a command line, change to directory *WBI_folder*/connectors/WebSphereMQWorkflow/utilities
   b. Enter the following:

```
FdlBorgen -isample_folder/WebSphereMQWorkflow_Samples.fdl -
osample_folder/SampleItem.in -nSampleItem -pMQWF_Structure_ -r3.1.0

FdlBorgen -isample_folder/WebSphereMQWorkflow_Samples.fdl -
osample_folder/SampleItemOrder.in -nSampleItemOrder -pMQWF_Structure_
-r3.1.0
```

3. **Load the sample business objects into the repository** Start IBM WebSphere ICS and, using Business Object Designer, select File"Open From File. Load the repository file named Sample_MQWF_Order_Objects.in (This file is located in the *WBI_folder*/connectors/WebSphereMQWorkflow/Samples folder.) Confirm that the sample business objects have been loaded.
4. **Load the sample collaboration template and collaboration objects into the repository** Using your WebSphere Business Integration System Manager, select select File"Open From File. Load the repository file named sample_MQWF_Order_Collaborations.in (located in the *WBI_folder*/connectors/WebSphereMQWorkflow/Samples folder).
5. **Compile the collaboration template** Using your WebSphere Business Integration System Manager, right-click the folder labeled Collaboration Templates and select Compile all from the drop down list.
6. **Restart IBM WebSphere ICS** Reboot InterChange Server to ensure that all changes take effect. Use System Monitor to ensure that all of the collaboration objects and connector controllers are in a green state.

Configuring the samples, template, adapter, and maps

This sections describes the samples as well as how to configure them and the adapter.

About the sample content

The samples are as follows:

- MQWF_DataStructure_SampleItemOrder This is a business object representing the data structure named SampleItemOrder in WebSphere MQ Workflow.
- MQWF_DataStructure_SampleItemOrder_Item This is a business object representing the child data structure Item contained in data structure SampleItemOrder in WebSphere MQ Workflow.
- MQWF_DataStructure_SampleItem This is a business object representing the data structure SampleItem in WebSphere MQ Workflow.
- MQWF_SampleItemOrder This is a container object for MQWF_DataStructure_SampleItemOrder. It holds an input and output data structure along with the various meta-objects used by the adapter. This is the object passed between the adapter and a collaboration when dealing with data structure SampleItemOrder.
- MQWF_SampleItem This is a container object for MQWF_DataStructure_SampleItem. It holds an input and output data structure along with the various meta-objects used by the adapter. This is the object passed between the adapter and a collaboration when dealing with data structure SampleItem.
- MO_MQWorkflow_ProcessInstance This object is used to track and control WebSphere MQ Workflow processes.
- MQWF_SampleItemRequest This is a container object for MQWF_DataStructure_SampleItemRequest and is used when WebSphere MQ Workflow sends a request to InterChange Server. It holds an input data structure along with various meta-objects used by the adapter. This is the object passed from the adapter to a collaboration when dealing with data structure SampleItemRequest.
- MQWF_SampleItemResponse This is a container object for MQWF_DataStructure_SampleItemRequest and is used when InterChange Server returns a response to WebSphere MQ Workflow. It holds an output data structure along with the various meta-objects used by the adapter. This is the object passed from a collaboration to the adapter when dealing with data structure SampleItemRequest.
- MQWF_GBO_SampleItem This is a container object for MQWF_DataStructure_SampleItem. It holds an input and output data structure and represents a fictitious generic business object in a collaboration.
- SampleItemOrderSync_MQWF_to_Port and SampleItemOrderSync_Port_to_MQWF These are collaboration objects used for exchanging business object MQWF_SampleItemOrder between the adapter and a Port Connector.
- SampleItemSync_MQWF_to_Port and SampleItemSync_Port_to_MQWFThese are collaboration objects used for exchanging business object MQWF_SampleItem between the adapter and a Port Connector.
- SampleWorkflowProcessControl_Port_to_MQWF This is a collaboration object used for retrieving and controlling business object MO_MQWorkflow_ProcessInstance
- SampleItemActivity_MQWF_to_MQWFThis is a collaboration object and is used for receiving MQWF_GBO_SampleItem from the adapter, passing it to a Port connector agent, and sending it back to the adapter.

- MQWF_Sample_RequesttoGBO This is a map that transfers data from a request business object MQWF_SampleItemRequest to a fictitious generic business object MQWF_GBO_SampleItem.
- MQWF_Sample_GBOtoResponse This is a map that transfers data from a fictitious generic business object MQWF_GBO_SampleItem to a response business object. MQWF_SampleItemResponse.

Copying the template class file

For the collaboration template to work, you must include the associate class file:

- Copy the files contained in *sample_folder*/classes to *WBI_folder*/collaborations/classes/UserCollaborations/classes

Note: Because the sample collaboration is based upon the CollaborationFoundation template (available separately), IBM does not provide the .CLM or .java components necessary for modifying the collaboration.

Configuring the connector

Configure connector properties as follows:

- AgentTraceLevel=3
- ApplicationPassword=(password for *username*)
- ApplicationUserName=*username*
- ArchiveQueue=MQWFCONN.ARCHIVE
- BOPrefix=MQWF_
- DeliveryTransport=IDL (optional)
- ErrorQueue=MQWFCONN.ERROR
- InputQueue=CWLDINPUTQ
- MQSeriesCCSID=*CCSID*
- MQSeriesChannel=*CHANNEL*
- MQSeriesHostname=*HOST*
- MQSeriesPort=*PORT*
- MQSeriesQueueManager=*queue manager*
- OutputQueue=FMC.FMCGRP.EXE.XML
- ReplyToQueue=MQWFCONN.REPLYTO
- UnsubscribedQueue=MQWFCONN.UNSUBSCRIBED

Supporting the sample business objects

To work with the sample business objects, you must make sure that the adapter supports them:

1. In System Manager, open the WBI Adapter for the WebSphere MQ Workflow and Port connector definitions
2. Click the Supported Business Objects tab, and add the following business objects:
   - MO_MQWorkflow_ProcessInstance
   - MO_MQWorkflow_ProcessTemplateConfig
   - MO_MQWorkflow_ContainerInfo
   - MO_MQWorkflow_ProcessInfo
   - MO_MQWorkflow_ActivityInfo

- MO_MQWorkflow_ActivityRequest
- MO_MQWorkflow_ActivityResponse
- MO_DataHandler_Default
- MO_DataHandler_DefaultXMLConfig
- MQWF_SampleItem
- MQWF_Structure_SampleItem
- MQWF_SampleItemOrder
- MQWF_Structure_SampleItemOrder
- MQWF_Structure_SampleItemOrder_Item
- MQWF_SampleItemRequest
- MQWF_SampleItemResponse
- MQWF_Structure_SampleItemRequest

Binding maps

To enable maps to transfer data, you must associate them with the adapter.

1. In System Manager, open the adapter definitions.
2. Click the Associated Maps tab.
3. Check Explicit Bindings for the following maps:
   - MQWF_Sample_RequesttoGBO
   - MQWF_Sample_GBOtoResponse

Final configuration steps

1. Reboot InterChange Server to ensure that all changes take effect.
2. Import *sample_folder*/WebSphereMQWorkflow_Samples.fdl to your IBM WebSphere MQ Workflow Runtime Server.

   Note: For information about importing the FDL file to Runtime, see I*BM WebSphere MQ Workflow: Getting Started with Buildtime*. If you are not concerned about overwriting existing runtime data, you can quickly load the sample workflows by opening a command line window and typing the following (WARNING: This will overwrite all runtime data):

   ```
   fmcibie /i=WebSphereMQWorkflow_Samples.fdl /u=ADMIN /y= /t
   /o (password: password)
   ```

Running the scenarios

Before you run the scenarios:

1. Start InterChange Server if it is not already running.
2. Start the WBI Adapter for WebSphere MQ Workflow if it is not already running, using the -fkey option (to prevent automatic polling).
3. Start the Visual Test Connector if it is not already running.

Simulate the Port Connector by starting the Visual Test Connector, defining a profile for PortConnector and binding the agent.

Synchronous request

This scenario passes business data to a defined workflow process and retrieves the end result. This is a synchronous call because, after issuing the request to WebSphere MQ Workflow, the adapter blocks until the initiated workflow process completes.

In this scenario, you retrieve the status of a fictitious order by passing an order key to workflow process `Lookup_Order_Status`. This workflow has only one action, which is to issue a retrieve to an IBM WebSphere Business Integration Server to get information about the order. This demonstrates how the adapter can issue a synchronous request to WebSphere MQ Workflow, and how WebSphere MQ Workflow can issue a synchronous request to the adapter.

1. **Create a sample order** Using the Visual Test Connector, create a new instance of business object `MQWF_SampleItemOrder` with the verb `Create` and populate it as follows (undefined values should be CxIgnore):
   `MQWF_SampleItemOrder`
   - `Input_ItemOrder`
     - `TrackingNumber = ABC123`
   - `MO_Config`
     - `ProcessTemplateName = Lookup_Order_Status`
     - `KeepName = false`
     - `UserId = UserName`
     - `ExecutionMode = Synchronous`
     - `ResponseTimeout = 600000`
     - `TimeoutFatal = false`
2. **Send this object to the adapter** The adapter will convert this object to a request message and issue it to the WebSphere MQ Workflow server. The adapter should not return immediately, but instead begin waiting for a response from WebSphere MQ Workflow.
   WebSphere MQ Workflow receives the request from the adapter to synchronously create and invoke process template `Lookup_Order_Status` using the data from object `MQWF_SampleItemOrder.Input_ItemOrder`. The first and only step of this workflow process is to retrieve data structure `SampleItemOrder` from the IBM WebSphere Business Integration Server using the tracking number as a key. To do this, MQ Workflow issues a request message to the input queue of the adapter and begins waiting for a response itself. This can be verified by checking the WebSphere MQ Workflow Client application.
3. Press p in the adapter agent window to poll for events. The adapter finds the request issued by WebSphere MQ Workflow (triggered by our original request). The adapter converts the request message to object `MQWF_SampleItemOrder` with verb `Retrieve` and posts it to collaboration `SampleItemOrderSync_MQWF_to_Port`.
4. Accept the request via the Visual Test Connector. Verify that attribute `TrackingNumber` of object `MQWF_SampleItemOrder.Input_ItemOrder` is `ABC123` (the same as that of our initial request). Populate object `MQWF_SampleItemOrder.Output_ItemOrder` as follows and then select `Reply Success` to complete the request:
   - `Output_ItemOrder`
     - `TrackingNumber = ABC123`
     - `Approved = YES`

   The adapter returns a response to WebSphere MQ Workflow, passing back the business data contained in

MQWF_SampleItemOrder.Output_ItemOrder.WebSphere MQ Workflow receives the response from the adapter and incorporates the data into a response message for the original request, which is then issued back to the ReplyTo queue of the adapter. The adapter retrieves the response message and returns any changes or errors to the collaboration. This completes the synchronous workflow request. The object returned to the collaboration should be populated as follows:

- MQWF_SampleItemOrder
  - Input_ItemOrder
    - TrackingNumber = ABC123
  - MO_Config
    - ProcessTemplateName = Lookup_Order_Status
    - KeepName = false
    - UserId = UserName
    - ExecutionMode = Synchronous
    - ResponseTimeout = 600000
    - TimeoutFatal = false
  - Output_ItemOrder
    - TrackingNumber = ABC123
    - Approved = YES
  - ProcessInstance This should be populated but the values cannot be predicted. The process state should be TERMINATED.

Note: If you fail to complete this process within 10 minutes (600000 milliseconds as configured in the meta-object), the adapter will report that it failed to receive a response from WebSphere MQ Workflow.

Asynchronous request

In this scenario, the adapter passers business data to a defined workflow process but does not wait for the process to complete. This is an asynchronous call— after issuing the request to WebSphere MQ Workflow, the adapter receives a process id to use in tracking the process as it executes in parallel. In this scenario, the adapter issues an order to workflow process Approve_Order to begin the task of approving an order (and later to check on whether the approval has completed). The workflow process retrieves information about the item ordered and updates the order approval based on whether sufficient quantities of the item are in stock. This scenario demonstrates how the adapter can trigger the start of workflow process asynchronously and how the adapter can monitor the status of a workflow process executing in parallel.

1. Using the Visual Test Connector, create a new instance of business object MQWF_SampleItemOrder with verb Create and populate it as follows (undefined values should be CxIgnore):
   - MQWF_SampleItemOrder
     - Input_ItemOrder
       - TrackingNumber = ABC123
       - Customer = Billy Bob
     - Item
       - Name = Hammer
       - Quantity = 1
     - MO_Config

- ProcessTemplateName = Approve_Order
- KeepName = false
- UserId = UserName
- ExecutionMode = Asynchronous
- ResponseTimeout = 500
- TimeoutFatal = false

2. Send MQWF_SampleItemOrder to the adapter, which converts this object to a request message and issues it to the WebSphere MQ Workflow server. The adapter then waits for a response that includes the process instance ID. WebSphere MQ Workflow receives the request from the adapter to asynchronously create and invoke process template Approve_Order using the data from object MQWF_SampleItemOrder.Input_ItemOrder. Once the process is started, WebSphere MQ Workflow immediately issues a response back to the adapter containing the id of the workflow process initiated. The workflow process also starts its first step, which is retrieving data structure SampleItem with name = Hammer from the IBM WebSphere Business Integration Server. These are two separate actions: a response is being issued to the ReplyTo queue of the adapter while simultaneously a request is being issued to the input queue of the adapter. The adapter receives the response from WebSphere MQ Workflow and returns a business object to the calling collaboration. This object is similar to the following:
   - MQWF_SampleItemOrder
     - Input_ItemOrder
       - TrackingNumber = ABC123
       - Customer = Billy Bob
     - Item
       - Name = Hammer
       - Quantity = 1
     - MO_Config
       - ProcessTemplateName = Approve_Order
       - KeepName = false
       - UserId = UserName
       - ExecutionMode = Asynchronous
       - ResponseTimeout = 5000
       - TimeoutFatal = false
     - ProcessInstance This should be populated but the values cannot be predicted.

       Note: Remember the value of attribute ProcessInstanceID for use later in this tutorial.

   At this point, the workflow process is executing in parallel to collaboration processing. The only means of tracking or controlling the workflow process is via the ProcessInstanceID returned in object MQWF_SampleItemOrder.ProcessInstance.
3. Using the Visual Test Connector, create a new instance of business object MO_MQWorkflow_ProcessInstance with verb Terminate. Using the XML API is required in WebSphere MQ Workflow 3.4 and is recommended for WebSphere MQ Workflow 3.3.2. To use the XML API, set Adapter Configuration Property JavaCorbaApi = False and, to monitor the workflow process

(MO_MQWorkflow_ProcessInstance), set ProcessInstanceName = *ProcInstName* (the *ProcInstName* returned in the previous step.

Note: For the XML API, the Restart and Delete verbs only are supported.

4. SendMO_MQWorkflow_ProcessInstance to the adapter and it should return the status of the workflow process. The attribute ProcInstState should equal RUNNING.
5. To resume the workflow process started in this scenario, press p in the adapter agent window to poll for events. The adapter finds the request issued by WebSphere MQ Workflow (triggered by the original request). The adapter then converts this request message to object MQWF_SampleItem with the verb Retrieve and posts it to collaboration SampleItemSync_MQWF_to_Port.
6. Accept the request via the Visual Test Connector. Verify that attribute Name of object MQWF_SampleItem.Input_Item is Hammer. Populate object MQWF_SampleItem.Output_Item as shown below and then select Reply Success to complete the request.
   - Output_Item
     - Name = Hammer
     - Price = 14.99
     - Stock = 20

   The adapter returns a response to WebSphere MQ Workflow, passing back the business data contained in MQWF_SampleItem.Output_Item. WebSphere MQ Workflow receives the response from the adapter and checks if the value of Stock is greater than the value Quantity in the original order. If so, there are enough hammers in stock to complete the order and therefore it is approved. The workflow process performs its final step by updating the order in InterChange Server and issues a SampleItemOrder data structure to the input queue of the adapter with the same key as the original order but now with attribute Approve equal to Y.
7. To process this final request from WebSphere MQ Workflow, press p in the adapter agent window to poll for events. The adapter finds the request issued by WebSphere MQ Workflow, creates object MQWF_SampleItemOrder with the verb Update and then posts the object to collaboration SampleItemOrderSync_MQWF_to_Port. You can simply accept this request (since there are no records to update in this scenario. Once the adapter issues a response, both the request from IBM WebSphere Business Integration Server and the workflow process are complete.

Workflow process control

This scenario demonstrates how to control a workflow process by terminating a process that is under way.

1. Start with step 1 of as described in "Asynchronous request" on page 113, but instead of issuing business object MO_MQWorkflow_ProcessInstance with verb Retrieve, change the verb to Suspend. Send this object to the adapter and verify via the WebSphere MQ Workflow Client application that the process suspends

   Note: The process will remain in a state of SUSPENDING until it can complete its first request to InterChange Server—this reflects the functionality of WebSphere MQ Workflow, and not the adapter.
2. Change the verb to Resume and re-send the object. The state of the workflow process changes back to RUNNING.

3. Change the verb to Terminate and re-send the object. The state of the workflow process will change to TERMINATED. You can verify this by issuing the object with verb Retrieve. In this manner, you have successfully controlled and monitored the state of a workflow process via ICS.

   Note: A request message may remain in the input queue of the adapter even thought the process that generated this request is terminated. This is normal. Although the adapter will process this request, WebSphere MQ Workflow will ignore any response generated.

Synchronous request from WebSphere MQ Workflow

This scenario simulates a synchronous request from WebSphere MQ Workflow to ICS and its response. The difference from Scenario 1 is that the adapter calls the collaboration asynchronously (whereas scenario 1 makes all calls synchronously). This scenario is more practical than scenario 1 because another workflow process does not have to wait for the workflow process to complete.

1. Create an update request. On the WebSphere MQ Workflow client, create and start a workflow process instance, and then fill an input data structure SampleItemRequest as follows:
   - SampleItemRequest
     - Name = Hammer
     - Price = 14.99
     - Stock = 20

   WebSphere MQ Workflow synchronously issues this request to the input queue of the adapter and waits for a response.
2. Enter p in the adapter agent window to poll for events. The adapter finds the request issued by WebSphere MQ Workflow. The adapter converts this request message to object MQWF_SampleItemRequest with verb Update. The adapter also converts the business object to a generic business object MQWF_GBO_SampleItem by using mapMQWF_Sample_RequesttoGBOand then publishes it asynchronously. At this point, the adapter does not wait for its response and can receive another request.
3. Accept the request via the Visual Test Connector. Collaboration SampleItemRequest_MQWF_to_MQWF subscribing the generic business object receives the object and it will be accepted by the Visual Test Connector.Verify that attribute Name of object MQWF_GBO_SampleItem.InputItem and MQWF_GBO_SampleItem.OutputItem is Hammer (the same as that of our initial request). Fills empty attributes of object MQWF_GBO_SampleItem.OutputItem as follows and then select ReplySuccess to send the generic business object to the adapter.
   - MQWF_GBO_SampleItemOrder
     - ContainerInfo
       - All attributes except ReturnCode are filled.
       - (Do not change ActImplCorrelID, which is used as ID in MQWF)
     - InputItem
       - Name = Hammer
       - Price = 14.99
       - Stock = 20
     - OutputItem
       - Name = Hammer

- Price = 11.25
- Stock = 8

The adapter receives the generic business object and converts it to object MQWF_SampleItemResponse by using map MQWF_Sample_GBOtoResponse. The adapter returns a response to WebSphere MQ Workflow with the business data contained in MQWF_SampleItemResponse.Output_Item. WebSphere MQ Workflow receives the response from the adapter and checks the value of ActImplCorrelID. If there is a workflow process that matches the value of ActImplCorrelID, then the process completes. The corresponding process instance in the WebSphere MQ Workflow client will disappear. (You may need to refresh the window.)

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.

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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

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

IIBM WebSphere Business Integration Adapter Framework V2.4.0.