IBM WebSphere Business Integration Adapters

Adapter for SunGard FRONT ARENA User Guide

Version 1.0.x

IBM WebSphere Business Integration Adapters

Adapter for SunGard FRONT ARENA User Guide

Version 1.0.x

Note!
Before using this information and the product it supports, read the information in "Notices" on page 169.

25June2004

This edition of this document applies to IBM WebSphere Business Integration Adapter for SunGard FRONT ARENA (Product ID 5724-i48), version 1.0.x, 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, legacy, and mainframe systems. The product set includes tools and templates for customizing, creating, and managing components for business process integration.

This document describes the installation, configuration, business object development, and troubleshooting for the IBM WebSphere Business Integration Adapter for SunGard® FRONT ARENA™.

Audience

This document is for consultants, developers, and system administrators who support and manage the WebSphere business integration system 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, and with the WebSphere MQ application.

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.

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

http://www.ibm.com/software/integration/websphere/support/. Select the component area of interest and browse the Technotes and Flashes sections.

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, filename, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
italic, *`italic`*	Indicates a variable name or a cross-reference.
blue 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 product is installed.

New in this release

New in release 1.0.x

June 2004:

Version 1.0.x is the first release of the Adapter for SunGard FRONT ARENA Adapter Guide.

Chapter 1. Overview

This chapter provides an overview, explaining terms you need to know, and describes adapter processing. It is important that you understand the adapter before installing, configuring, and using it.

Task roadmap

To use the Adapter FRONT ARENA, you must perform the tasks described in Table 1.

Table 1. Using the adapter: Task roadmap

Task	Associated procedure (see...)	For more information (see...)
Installing the connector	Chapter 2, "Installing the adapter," on page 15	*Installation Guide for WebSphere Business Integration Adapters*
Configuring business and meta- objects	Chapter 3, "Business objects for FRONT ARENA," on page 31	*Business Object Development Guide*
Configuring data handlers	Chapter 4, "Configuring the data handler," on page 35	*Data Handler Guide*
Configuring the connector	Chapter 5, "Configuring the connector," on page 41	Chapter 6, "Configuring the bridge," on page 57, Chapter 7, "Configuring the application," on page 71, Appendix B, "Connector specific properties," on page 125, and *Connector Development Guide*
Running the connector	Chapter 8, "Running the adapter," on page 73	*Connector Development Guide*
Maintaining the connector	Chapter 9, "Troubleshooting the adapter," on page 99	

Terminology

To understand the adapter, you must know these terms:

adapter
The component in the WebSphere business integration system that provides components to support communication between an integration broker and either an application or a technology. An adapter always includes a connector, message files, and configuration tools. It can also include an Object Discovery Agent (ODA) or a data handler.

adapter framework
The software that IBM provides to configure and run an adapter. The

runtime components of the adapter framework include the Java runtime environment, the connector framework, and the Object Discovery Agent (ODA) runtime. This connector framework includes the connector libraries (C++ and Java) needed to develop new connectors. The ODA runtime includes the library in the Object Development Kit (ODK) needed to develop new ODAs. The configuration components include the following tools:

- Business Object Designer,
- Connector Configurator,
- Log Viewer,
- System Manager,
- Adapter Monitor,
- Test Connector
- and, optionally, any Object Discovery Agents (ODAs) associated with an adapter.

Adapter Development Kit (ADK)
A development kit that provides some examples for adapter development, including example connectors and Object Discovery Agents (ODAs).

Arena Message Broker (AMB)
The FRONT ARENA AMB is a message oriented middleware with store-and-forward capabilities that allow internal and external applications to communicate with each other based on a publish/subscribe protocol.

connector
The component of an adapter that uses business objects to send information about an event to an integration broker (event notification) or receive information about a request from the integration broker (request processing). A connector consists of the connector framework and the connector's application-specific component.

connector framework
The component of a connector that manages interactions between a connector's application-specific component and the integration broker. This component provides all required management services and retrieves the metadata that the connector requires from the repository. The connector framework, whose code is common to all connectors, is written in Java and includes a C++ extension to support application-specific components written in C++.

connector controller
The subcomponent of the connector framework that interacts with collaborations. A connector controller runs within InterChangeServer and initiates mapping between application-specific and generic business objects, and manages collaboration subscriptions to business object definitions.

integration broker
The component in the WebSphere business integration system that integrates data among heterogeneous applications. An integration broker typically provides a variety of services that include: the ability to route data, a repository of rules that govern the integration process, connectivity to a variety of applications, and administrative capabilities that facilitate integration.

WebSphere business integration system
An enterprise solution that moves information among diverse sources to perform business exchanges, and that processes and routes information

among disparate applications in the enterprise environment. The business integration system consists of an integration broker and one or more adapters.

WebSphere MQ Integrator Broker, Version 2.2
A message broker product that transforms and routes messages between WebSphere MQ queues. The technology enables applications to communicate asynchronously by delivering messages to and receiving messages from potentially remote queues. A major change with MQ Integrator Broker is the addition of message flows that add the ability to format, store, and route messages based on user-defined logic.

Adapter architecture

The adapter is metadata-driven. The adapter uses an MQ implementation of the Java™ Message Service (JMS), an API for accessing enterprise-messaging systems.

The Adapter for FRONT ARENA uses WebSphere MQ queues to enable asynchronous data exchange from FRONT ARENA to the broker and from the broker to FRONT ARENA. Data is sent between the queues for FRONT ARENA and the broker in the form of text messages. A FRONT ARENA data handler is used to convert the data into business objects that can be processed by any broker.

For information about using the adapter in synchronous exchanges, see "Synchronous request and reply interactions" on page 5.

The adapter enables the WebSphere MQ Adapter to exchange messages with the FRONT ARENA application, version 1.5.

The Adapter for FRONT ARENA can be used in a solution that uses the integration broker to integrate business data exchanges between FRONT ARENA and other enterprise information system applications for which appropriate adapters have been installed.

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

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

This document contains information about the adapter framework and the connector. It refers to both of these components as the adapter. For more information about the relationship of the integration broker to the adapter, see the documentation for *IBM WebSphere InterChange Server System Administration Guide*.

Note: All WebSphere business integration adapters operate with an integration broker. The Adapter for FRONT ARENA operates with the InterChange Server integration broker, which is described in the *Technical Introduction to IBM WebSphere InterChange Server*.

Asynchronous messages from FRONT ARENA to the integration broker

When a trade is placed in FRONT ARENA, an INSERT_TRADE message is generated in native FRONT ARENA message format and is placed in the WebSphere MQ output queue as shown in figure Figure 1. (In this illustration, it is assumed that FRONT ARENA and the integration broker are installed on different machines using different queue managers, making it necessary to have a remote queue definition for output from FRONT ARENA, connecting to an input queue that is local to the integration broker. A single queue functions as both the output queue from FRONT ARENA and input queue for the integration broker.)

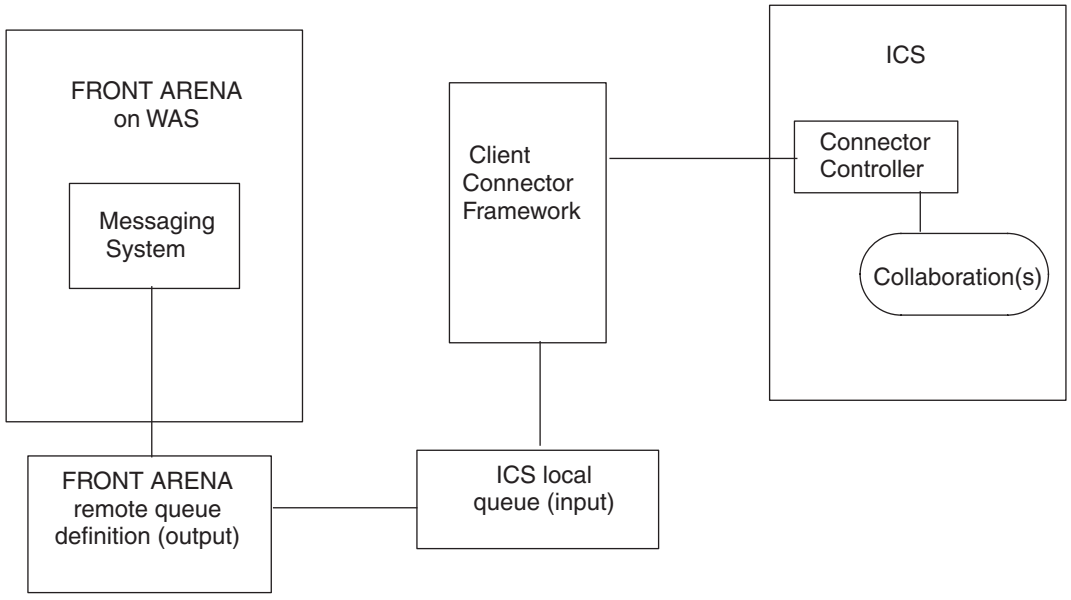

Figure 1. Adapter architecture

To detect data events in FRONT ARENA, the adapter polls the FRONT ARENA output queues for new messages. When it discovers a new message, the adapter passes it to an input queue, calls a data handler to convert the message to a business object that is specific to the structure of data originating from FRONT ARENA, and then passes the business object to the connector in InterChange Server. The connector invokes maps to generate a generic business object from the FRONT ARENA-specific business object, and then delivers the generic business object to one or more collaboration objects. After the collaboration objects have processed the business object, the generic business object is mapped to an application-specific business object, which is delivered to an adapter (such as a WebSphere Business Integration adapter for SAP) that has been configured for a back-end application.

The following picture visualizes how FRONT ARENA is integrated in the WBI context by means of the bridge, the connector, and WebSphere MQ.

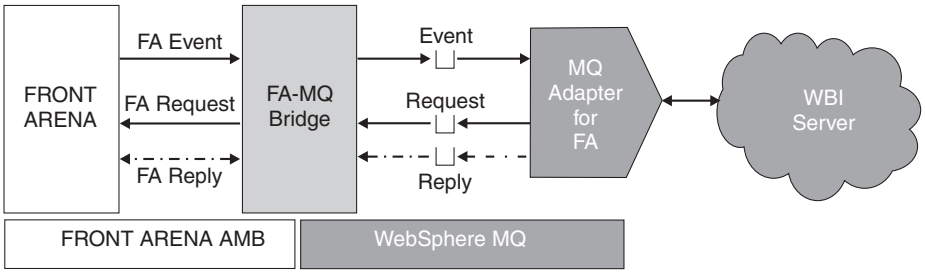

Figure 2. High-level architectural diagram

Asynchronous messages to FRONT ARENA from InterChange Server

In the opposite direction, the Adapter for FRONT ARENA receives business objects from collaborations, converts them into messages using the data handler and delivers the messages to the FRONT ARENA WebSphere MQ queue.

Synchronous request and reply interactions

Synchronous request and reply interactions require additions to or customization of the FRONT ARENA application, as described in the following topics.

Requests from FRONT ARENA to InterChange Server

You can use the adapter to set up a synchronous message flow for request and reply interactions from the FRONT ARENA messaging system to InterChange Server or other external systems. For more information about this approach, see the integration documentation for FRONT ARENA, version 1.5, and the IBM WebSphere business integration system.

Requests from InterChange Server to FRONT ARENA

Note: This approach requires a customization of the commands that are executed in FRONT ARENA when a business object comes from InterChange Server. The commands should retrieve the `ReplyTo` queue from the message, and place a reply on the queue within the `ResponseTimeout` interval. For information about creating and customizing commands in FRONT ARENA, refer to the *Programmer's Guide for FRONT ARENA, 1.5. 5.4*.

Synchronous versus asynchronous processing

The bridge component of the adapter can be configured to handle requests either synchronously or asynchronously. In the asynchronous case, the request is simply forwarded to the ARENA Message Broker (AMB). Whether or not the request can be fulfilled is not verified. No reply is sent back to the originator of the request.

In synchronous mode, the bridge component of the adapter checks the outcome of the request processing and returns this to the application that originally issued the request. Since the AMB is an asynchronous messaging system, a client application does not get a synchronous reply to a request. The response can arrive at any later point in time, and the client application must relate it to the corresponding request.

The AMB does not have in-built request-reply correlation support. Therefore, the correlation can only be done based on elements of the payload of the request and reply message.

Event notification

Notification of data events that have occurred in the FRONT ARENA application is accomplished through the polling mechanism of the adapter. The adapter can poll multiple input queues, polling each in a round-robin manner and retrieving a specified number of messages from each queue. For each message retrieved during polling, the adapter adds a dynamic child meta-object (if specified in the business object). The child meta-object values can direct the adapter to populate attributes with the format of the message as well as with the name of the input queue from which the message was retrieved.

When a message is retrieved from the input queue, the adapter looks up the business object name associated with the `FORMAT` field contained in the message header. The message body, along with a new instance of the appropriate business object, is passed to the data handler. If a business object name is not associated with the format, the message body alone is passed to the data handler. If a business object is successfully populated with message content, the adapter checks to see if it is subscribed, and delivers it to InterChange Server.

Business objects and WebSphere MQ message header

The type of business object and verb used in processing a message is based on the `FORMAT` field contained in the WebSphere MQ message header. The adapter uses meta-object entries to determine the business object name and verb. You construct a meta-object to store the business object name and verb to associate with the WebSphere MQ message header `FORMAT` field text.

Optionally, you can construct a dynamic meta-object that is added as a child to the business object that is passed to the adapter. The child meta-object values override those specified in the static meta-object that is specified for the adapter as a whole. If the child meta-object is not defined or does not define a required conversion property, by default, the adapter examines the static meta-object for the value. You can specify one or more dynamic child meta-objects instead of, or to supplement, a single static adapter meta-object.

Application-to-adapter communication

The Adapter for FRONT ARENA allows the WebSphere InterChange Server to exchange business objects with the FRONT ARENA application.

The Adapter for FRONT ARENA consists of the WebSphere MQ Adapter, the bridge component, and the data handler. The adapter makes use of the WebSphere MQ implementation of the Java Message Service (JMS). JMS is an open-standard API for accessing enterprise messaging systems. It is designed to allow business applications to asynchronously send and receive business data and events.

The bridge component of the adapter forwards messages from the FRONT ARENA AMB messaging system to the WebSphere MQ messaging system and vice versa. It does not deal with the content of messages. No message content transformation is performed; only messaging system dependent headers are exchanged. As the messages, or more accurately, the payload of the messages, originate from or eventually end up in the FRONT ARENA system, they have the native FRONT ARENA format, in either case. The application communicating with FRONT ARENA by way of the adapter must be able to cope with this format. The messages on the queue can then be accessed by either the WebSphere MQ Adapter or other third party applications.

The FRONT ARENA application has a middleware layer of its own, the ARENA Message Broker (AMB). This is an asynchronous messaging system following the publish and subscribe paradigm. External applications are recommended to access FRONT ARENA through the APIs provided by the AMB. An API is provided for accessing and utilizing the AMB. This is the point of integration recommended for any applications electronically feeding data (for example, trades) or extracting data.

The API can be used for synchronizing a FRONT ARENA system with other financial market applications or vice versa. If FRONT ARENA is set up correspondingly, any newly created FRONT ARENA objects, or changes to existing FRONT ARENA objects, for example, trades, positions, are published on the AMB. External applications can register to the AMB and subscribe to all or subsets of these notifications. In the opposite direction, external applications can create new FRONT ARENA objects or change existing ones by putting appropriate messages on the AMB.

The AMB APIs are low level C interfaces and extensive memory management is required around API calls. The bridge component of the adapter connects to the FRONT ARENA AMB on one side, and by way of the WebSphere MQ queues - to the adapter on the other side. The bridge component of the adapter subscribes to create, update, and delete notifications for a configurable set of FRONT ARENA objects, and passes all notification event messages received from FRONT ARENA to the WebSphere MQ queue. The bridge also forwards service request messages received from the adapter to FRONT ARENA, and optionally returns the results of the requests to the adapter. The data handler component of the adapter copes with the native FRONT ARENA message formats.

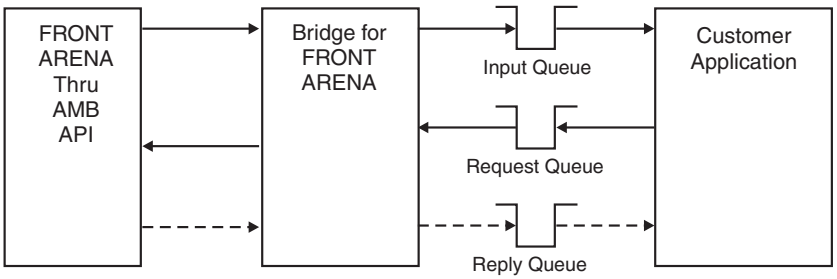

Figure 3. Direct use of the bridge component

The bridge can be used in two contexts:

- By customer applications that want to communicate directly with FRONT ARENA by means of WebSphere MQ messages.
- In WebSphere Business Integration (WBI) based business integration scenarios, in conjunction with the Adapter for FRONT ARENA.

Message request

Figure 4 illustrates a message request communication. When the `doVerbFor()` method receives a business object from a collaboration, the adapter passes the business object to the data handler. The data handler converts the business object into a FRONT ARENA native text message and the adapter issues it as a message to a queue. There, the JMS layer makes the appropriate calls to open a queue session and route the message.

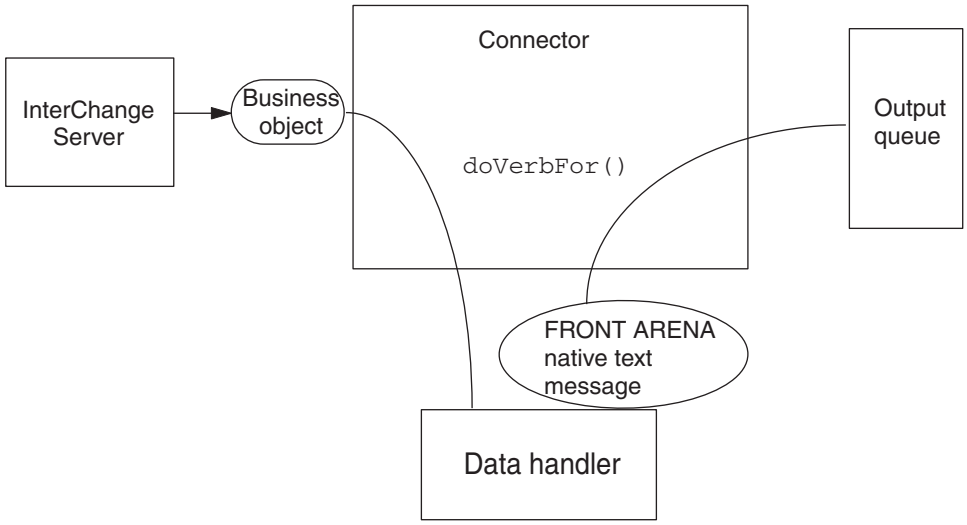

Figure 4. Application-to-adapter communication method: Message request

Message return

Figure 5 illustrates the message return direction. The `pollForEvents()` method retrieves the next applicable message from the input queue. The message is staged in the in-progress queue where it remains until processing is complete. Using either the static or dynamic meta-objects, the adapter first determines whether the message type is supported. If so, the adapter passes the message to the configured data handler, which converts the message into a business object. The verb that is set reflects the conversion properties established for the message type. The adapter then determines whether the business object is subscribed to by a collaboration. If so, the `gotApplEvents()` method delivers the business object to InterChange Server, and the message is removed from the in-progress queue.

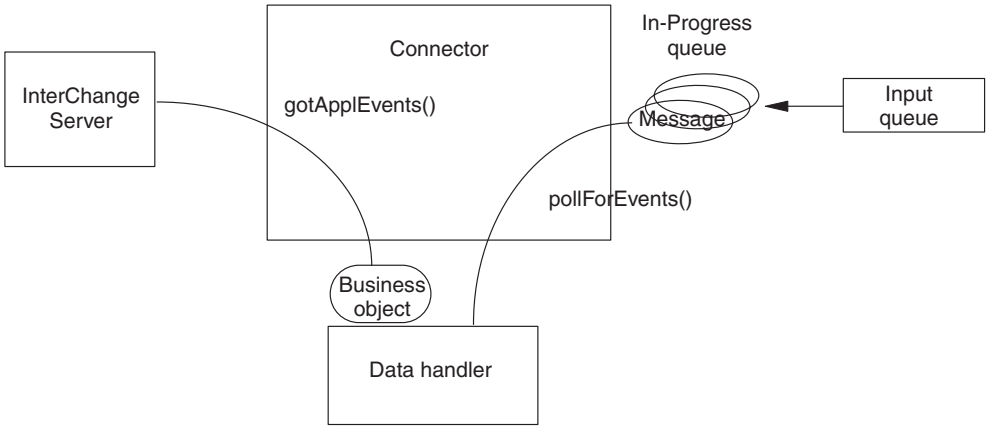

Figure 5. Application-to-adapter communication method: Message return

Event notification

For event notification, the adapter detects events written to the FRONT ARENA AMB. For each newly created or modified object, an event notification message is published on the AMB under a predefined topic that contains the type of the affected object. Event processing is asynchronous, FRONT ARENA messages are put on the queue without waiting for a response.

AMB client applications can subscribe to all messages published under certain topics. This way, they become aware of changes in the FRONT ARENA system. Based on this mechanism, other financial market applications can be kept in synchrony with FRONT ARENA in a scenario where the FRONT ARENA system is the master application, for example. The bridge component of the Adapter for FRONT ARENA acts as an AMB client application. It does not use the information received from FRONT ARENA by itself, but passes it on to a third party by way of the WebSphere MQ queue. The types of events subscribed to, and forwarded, by the bridge depend on its configuration. These settings must be consistent with the corresponding FRONT ARENA configuration parameters, and more precisely, the corresponding FRONT ARENA AMBA parameters.

Message retrieval

The adapter uses the `pollForEvents()` method to poll the queue for messages at regular intervals. When the adapter finds a message, it retrieves it from the queue and examines it to determine its format. If the format has been defined in the adapter's static meta-object, the adapter passes both the message body and a new instance of the business object associated with the format to the FRONT ARENA data handler. The data handler is expected to populate the business object. If the format is not defined in the static meta-object, the adapter passes only the message body to the data handler; the data handler is expected to determine, create, and populate the correct business object for the message. See "Error handling" on page 101 for event failure scenarios.

The adapter processes messages by first opening a transactional session to the input queue. The adapter then moves all messages to an in-progress queue. The message is held in the in-progress queue until processing is complete. If the adapter shuts down unexpectedly during processing, the message remains in the in-progress queue instead of being reinstated to the original input queue.

Note: Transactional sessions with WebSphere MQ require that every requested action on a queue be performed and committed before events are removed from the queue. Accordingly, when the adapter retrieves a message from the queue, it does not commit to the retrieval until three things occur:

1. The message has been converted to a business object.
2. The business object is delivered to InterChange Server by the `gotApplEvents()` method.
3. A return value is received.

Message recovery

Upon initialization, the adapter 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.

Fail on startup

With the fail on startup option, if the adapter 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 a different queue.

Reprocessing messages

With the reprocessing option, if the adapter 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 adapter begins processing messages from the input queue.

Ignoring messages

With the ignore option, if the adapter finds any messages in the in-progress queue during initialization, the adapter ignores them, but does not shut down.

Logging errors

With the log error option, if the adapter finds any messages in the in-progress queue during initialization, it logs an error but does not shut down.

Message archiving

If the connector property, `ArchiveQueue`, is specified and identifies a valid queue, the adapter places copies of all successfully processed messages in the archive queue. If `ArchiveQueue` is undefined, messages are discarded after processing. For more information on archiving unsubscribed or erroneous messages, see "Error handling" on page 101.

Note: By JMS conventions, a retrieved message cannot be issued immediately to another queue. To enable archiving and re-delivery of messages, the adapter first produces a second message that duplicates the body and the header (as applicable) of the original. To avoid conflicts with the FRONT ARENA messaging service, only JMS-required fields are duplicated. Accordingly, the `FORMAT` field is the only additional message property that is copied for messages that are archived or re-delivered.

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 JMS queue. To become JMS-enabled, you must configure the connector `DeliveryTransport` standard property to JMS. Once 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 78.

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

Business object requests

Business object requests are processed when InterChange Server sends a business object to the `doVerbFor()` method. Using the configured data handler, the adapter converts the business object to an WebSphere MQ message and issues it. There are no requirements regarding the type of business objects processed except those of the data handler.

Verb processing

The adapter processes business objects passed to it by a collaboration based on the verb for each business object. The adapter uses business object handlers and the `doForVerb()` method to process the business objects that the adapter supports. The adapter supports the following business object verbs:

- Create
- Update
- Delete
- Retrieve

Note: Business objects with Create, Update, and Delete verbs can be issued either asynchronously or synchronously. The default mode is asynchronous. The adapter does not support asynchronous delivery for business objects with the Retrieve, Exists, or Retrieve by Content verbs, Accordingly, for Retrieve, Exists, or Retrieve by Content verbs, the default mode is synchronous.

Create, update, and delete verbs

The processing of business objects with the create, update, and delete verbs will depend on whether the objects are issued asynchronously or synchronously.

Asynchronous message delivery

The default delivery mode for business objects with create, update, and delete verbs is asynchronous. A message is created from the business object by using a data handler and then it is written to the output queue. If the message is delivered, the adapter returns BON_SUCCESS, else BON_FAIL.

Note: The adapter has no way of verifying whether the message is received or if action has been taken.

Synchronous message delivery

Note: This approach requires a customization of the commands that are executed in the FRONT ARENA application when a business object comes from the InterChange Server. The commands should retrieve the `ReplyTo` queue from the message, and place a reply on the queue within the ResponseTimeout interval. For information about creating and customizing commands in FRONT ARENA, refer to the *Programmer's Guide for SunGard FRONT ARENA, 1.5, 5.4*.

If a `replyToQueue` has been defined in the connector properties and a `ResponseTimeout` exists in the conversion properties for the business object, the adapter issues a request in synchronous mode. The adapter then waits for a response to verify that appropriate action was taken by FRONT ARENA.

Note: MQMDs are message descriptors. MQMDs contain the control information accompanying application data when a message travels from one application to another. You must specify a value for the MQMD attribute, `OutputFormat`, in either your static or dynamic meta-object.

The adapter initially issues a message with a header as shown in the following table.

Table 2. Request message descriptor header (MQMD)

Field	Description	Value
Format	The format name.	The output format as defined in the conversion properties where the output is truncated to 8 characters to meet IBM requirements (for example: `MQSTR`).
MessageType	The message type.	`MQMT_DATAGRAM`*
Report	The options for the report message requested.	When a response message is expected, this field is populated as follows: `MQRO_PAN`*, to indicate that a positive-action report is required if processing is successful. `MQRO_NAN`*, to indicate that a negative-action report is required if processing fails. `MQRO_COPY_MSG_ID_TO_CORREL_ID`*, to indicate that the correlation ID of the report generated should equal the message ID of the request originally issued.
ReplyToQueue	The name of a reply queue.	When a response message is expected, this field is populated with the value of the connector property `ReplyToQueue`.
Persistence	The message persistence.	`MQPER_PERSISTENT`*
Expiry	The message lifetime.	`MQEI_UNLIMITED`*

* Indicates a constant that is defined by IBM.

The message header described in Table 2 is followed by the message body. The message body is a business object that has been serialized using the data handler.

The Report field is set to indicate that both positive and negative action reports are expected from FRONT ARENA. The thread that issued the message waits for a response message that indicates whether FRONT ARENA was able to process the request.

When FRONT ARENA receives a synchronous request from the adapter, it processes the data of the business object and issues a report message as described in the following tables.

Table 3. Response message descriptor header (MQMD)

Field	Description	Value
Format	The format name.	The input format of `busObj` as defined in the conversion properties.
MessageType	The message type.	`MQMT_REPORT`*

* Indicates a constant that is defined by IBM.

Table 4. Population of response messages

Verb	Feedback field	Message body
Create, Update, or Delete	SUCCESS VALCHANGE	(Optional) A serialized business object reflecting changes.
	VALDUPES FAIL	(Optional) An error message.

Table 5. WebSphere MQ feedback codes and InterChange Server response values

WebSphere MQ feedback code	Equivalent InterChange Server responseInterChange Server*
`MQFB_NONE` (this is the default if no feedback code is specified)	VALCHANGE
`MQFB_PAN` or `MQFB_APPL_FIRST`	SUCCESS
`MQFB_NAN` or `MQFB_APPL_FIRST + 1`	FAIL
`MQFB_APPL_FIRST + 2`	VALCHANGE
`MQFB_APPL_FIRST + 3`	VALDUPES
`MQFB_APPL_FIRST + 4`	MULTIPLE_HITS
`MQFB_APPL_FIRST + 5`	FAIL_RETRIEVE_BY_CONTENT
`MQFB_APPL_FIRST + 6`	BO_DOES_NOT_EXIST
`MQFB_APPL_FIRST + 7`	UNABLE_TO_LOGIN
`MQFB_APPL_FIRST + 8`	APP_RESPONSE_TIMEOUT (results in immediate termination of connector)
`MQFB_NONE`	What the connector receives if no feedback code is specified in the response message

If the business object can be processed, the application creates a report message with the feedback field set to `MQFB_PAN` (or a specific InterChange Server value). Optionally, the application populates the message body with a serialized business object containing any changes. If the business object cannot be processed, the application creates a report message with the feedback field set to `MQFB_NAN` (or a specific InterChange Server value), then, optionally includes an error message in the message body. In either case, the application sets the `correlationID` field of the message to the `messageID` of the adapter message and issues it to the queue specified by the `replyTo` field.

Upon retrieval of a response message, the adapter matches the `correlationID` of the response to the `messageID` of a request message. The adapter then notifies the thread that issued the request. Depending on the feedback field of the response, the adapter either expects a business object or an error message in the message body. If a business object was expected, but the message body is not populated, the adapter simply returns the same business object that was originally issued by

InterChange Server for the `Request` operation. If an error message was expected, but the message body is not populated, a generic error message will be returned to InterChange Server along with the response code.

Custom feedback codes: You can extend the WebSphere MQ feedback codes to override the default interpretations, shown in Table 5 on page 13, by specifying the connector property, `FeedbackCodeMappingMO`. This property allows you to create a meta-object in which all InterChange Server-specific return status values are mapped to the WebSphere MQ feedback codes.

The return status assigned to a feedback code is passed to InterChange Server. For more information, see "FeedbackCodeMappingMO" on page 76.

Retrieve verb

For business objects with the retrieve verb support synchronous delivery only, the connector processes business objects with these verbs as it does for the synchronous delivery defined for create, update and delete. However, when using retrieve, exists, and retrieve by content verbs, the `responseTimeout` and `replyToQueue` are required. Furthermore, for retrieve and retrieve by content verbs, the message body must be populated with a serialized business object to complete the transaction.

Table 6 shows the response messages for these verbs.

Table 6. Population of response message

Verb	Feedback field	Message body
Retrieve	FAIL	(Optional) An error message.

Data handler message processing

The Adapter for FRONT ARENA uses a newly developed data handler which is capable of coping with native FRONT ARENA messages. The data handler supports both message-to-business object conversion and conversion in the opposite direction.

When converting messages to business objects, the data handler relies on a newly developed name handler which determines the type of business object to be created from certain message fields. For details about this and other connector-specific properties, see Chapter 4, "Configuring the data handler," on page 35. For details about developing a data handler, see the *Data Handler Guide*.

Chapter 2. Installing the adapter

This chapter describes how to install the Adapter for FRONT ARENA.

Adapter environment

Before installing, configuring, and using the adapter, you must understand its environment requirements.

Adapter components

The Adapter for FRONT ARENA ships with the following parts:

- The *Adapter for SunGard FRONT ARENA User Guide* (which should be read in conjunction with the documentation for the *Adapter for WebSphere MQ User Guide*.
- The adapter's data handler related meta-objects: MO_DataHandler_FrontArenaDefaultConfig.xsd, MO_DataHandler_FrontArenaNativeConfig.xsd, and MO_FrontArena_DefaultConfig.xsd.
- The special data handler used by the adapter which comes as a JAR file, FrontArenaDataHandler.jar.
- An example application-specific business object definition for FA_trade, with its entire child application-specific business objects (FA_trade.xsd).

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 1.0 of the Adapter for FRONT ARENA 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 the *Release Notes* for any exceptions.

Note: For instructions on installing your integration broker and its prerequisites, see the following guides:
For WebSphere InterChange Server, 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

This adapter is supported on the following software:

Operating systems:

One of the following application platforms:

- AIX 5.1, AIX 5.2
- Solaris 8.0
- HP UX 11.0, HP UX 11i
- Windows 2000

Third-party software:

- FRONT ARENA version 1.5

Adapter dependencies

The Adapter for FRONT ARENA requires the WebSphere Application Server libraries and API.

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 107

Globalization

Globalization is supported as far as it is supported by the standard WebSphere MQ technology adapter.

Prerequisite tasks

The following steps must be made to convert the pre-installed WebSphere MQ adapter to a FRONT ARENA adapter.

These tasks are:

1. Load the following application-specific business object definitions:
   - `FA_trade.xsd`
   - `MO_DataHandler_FrontArenaDefaultConfig.xsd`, `MO_DataHandler_FrontArenaNativeConfig.xsd` (from `MO_DataHandler_FrontArenaDefaultConfig.xsd`), and `MO_FrontArena_DefaultConfig.xsd`
2. Set the connector's standard property, `ApplicationName`, to the name assigned to the adapter (for example, `FrontArenaConnector`).
3. Set the connector's application configuration properties `DataHandlerConfigMO`, `DataHandlerMIMIType`, and `ConfigurationMetaObject` as described in "Default adapter configuration properties" on page 27.
4. Define the objects imported in step 1 as supported business objects of the adapter (with Agent Support on).
5. Copy the adapter's message file to the `messages` subdirectory of the `connectors` directory of the WebSphere Interchange Server installation.
6. Copy the adapter's data handler JAR file to the `Datahandlers` subdirectory of the `connectors` directory of the WebSphere Interchange Server installation.

Configuring WebSphere MQ queues

The WebSphere MQ queue configuration required for using the adapter depends in part upon the topology of your FRONT ARENA and IBM WebSphere InterChange Server installations. You may be using any of the following topologies:

- Single machine

  FRONT ARENA, IBM WebSphere InterChange Server, and the adapter are all installed on the same machine.
- Two machines with two queue managers

  FRONT ARENA is installed on one machine, and IBM WebSphere InterChange Server and the connector are installed on another machine. A different queue manager is used on each machine.
- Two machines with a single queue manager

  FRONT ARENA is installed on one machine, and IBM WebSphere InterChange Server and the connector are installed on another machine. The same queue manager is used to manage the queues on both machines.

Single-machine topology

In this topology, FRONT ARENA, InterChange Server, and the Adapter for FRONT ARENA are all installed on a single machine. A single queue manager handles all the WebSphere MQ queues used in the solution. It is recommended that you use the queue manager that you set up when you installed InterChange Server.

This topology requires queues that perform the following roles:

- Inbound queue

  FRONT ARENA requires that this queue exist. However, the adapter does not make use of the queue in this solution.
- Parallel inbound queue

FRONT ARENA requires that this queue exist. However, the adapter does not make use of the queue in this solution.

- Serial inbound queue

  This is for receiving messages sent from InterChange Server to FRONT ARENA.
- Outbound queue

  This is for sending messages from FRONT ARENA to InterChange Server.
- In progress queue

  The original versions of valid messages sent from FRONT ARENA to InterChange Server are stored here until the adapter completes processing; when processing is completed, the original message is moved to the local archive queue.
- `Archive_Queue`

  When a message has been fully processed by the adapter and sent to InterChange Server from FRONT ARENA, the original version of the message is stored here.
- `Unsubscribe_Queue`

  If a message is successfully parsed but does not correspond to any business object supported by the adapter, it is stored here.
- `ICS_Error_Queue`

  If a message is not successfully converted to a business object and sent to the InterChange Server, it is stored here.
- `WCS_Error_Queue`

  Stores messages that cannot be successfully processed by FRONT ARENA.
- `ReplyToQueue`

  Used only in configurations that are set up for synchronous data exchange.

All of the above queues are local in the single machine topology. If you create the queues manually, you choose the names that you assign for them; if you use the batch file provided with this solution (described below), the batch file will create the queues with pre-assigned names.

If you are using the adapter in a Windows environment, you can use a batch file to generate queues that are appropriate for a single machine topology. The file is installed with the product package in the `\Connector\WebSphereCommerce\Utilities` subdirectory within the root directory used for the IBM InterChange Server installation. To create the queues using the batch file, run the file `ConfigureWebSphereCommerceAdapter.bat`, as follows:

From the command prompt, enter:

```
ConfigureWebSphereCommerceAdapter
<InterChangeServerName>.queue.manager
```

Where, `<InterChangeServerName>` is the name of your WebSphere InterChange Server.

This will create a queue manager named `InterChange` *`ServerName`*`.queue.manager` and it will create the required WebSphere MQ queues. The names of the created queues, as created by the batch file, can be seen in the following table.

Table 7. WebSphere MQ queues

WebSphere MQ queue name	Description
WC_MQCONN.IN_PROGRESS	The in-progress queue for the adapter.
WC_MQCONN.ERROR	The InterChange Server error queue for the adapter.
WC_MQCONN.ARCHIVE	The archive queue for the adapter.
WC_MQCONN.REPLY	The reply-to queue for the adapter.
MC_MQCONN.UNSUBSCRIBED	The unsubscribed queue for the adapter.
WCS_Serial_Inbound	The serial inbound queue for FRONT ARENA. This queue name must match a JMS queue name defined for FRONT ARENA, as described in "Configuring WebSphere Application Server JMS settings" on page 22.
WCS_Outbound	An outbound queue for FRONT ARENA. This queue name must match a JMS queue name defined for FRONT ARENA, as described in "Configuring WebSphere Application Server JMS settings" on page 22.
WCS_Parallel_Inbound	An inbound parallel queue for FRONT ARENA. This queue name must match a JMS queue name defined for FRONT ARENA, as described in "Configuring WebSphere Application Server JMS settings" on page 22.
WCS_Error	An error queue for FRONT ARENA. This queue name must match a JMS queue name defined for FRONT ARENA, as described in "Configuring WebSphere Application Server JMS settings" on page 22.
WCS_Inbound	An inbound queue for FRONT ARENA. This queue name must match a JMS queue name defined for FRONT ARENA, as described in "Configuring WebSphere Application Server JMS settings" on page 22.

Two-machine, two-Queue Manager topology

In this topology, FRONT ARENA is installed on one machine and IBM WebSphere InterChange Server and the Adapter for FRONT ARENA are installed on another machine.

WebSphere MQ must be installed on each machine, and each installation uses a different queue manager. These are the queues you create on each machine.

InTable 8 on page 20, the queue names indicate the role of each queue, but you can establish different queue names, as long as you synchronize the queue names with JMS queue names used on your FRONT ARENA system.

In the table, the prefix WCS indicates a queue that is created on the machine on which the FRONT ARENA system is installed, and that is managed by a queue manager that resides on that machine. The prefix ICS indicates a queue that is created on the machine on which InterChange Server and the connector are installed, and the queue is managed by a queue manager that resides on that machine.

Table 8. FRONT ARENA and InterChange Server queue names

Queues on the FRONT ARENA machine	Queues on the InterChange Server machine
WCS_Outbound Queue	**ICS_Inbound queue**
For sending messages from FRONT ARENA to InterChange Server. This queue is created as a remote queue definition, pointing to ICS_Inbound on the InterChange Server machine as the remote queue.	For receiving messages sent from FRONT ARENA to InterChange Server.
WCS_Serial Inbound queue	**ICS_Outbound queue**
For receiving messages sent from InterChange Server to FRONT ARENA.	For sending messages from InterChange Server to FRONT ARENA. This queue is created as a remote queue definition, pointing to `WCS_Serial Inbound` on the FRONT ARENA machine as the remote queue.
To InterChange Server	**To FRONT ARENA**
Transmission queue from FRONT ARENA system to InterChange Server.	Transmission queue from InterChange Server to the FRONT ARENA system.
WCS_Error_queue	**ICS_Error_queue**
Stores messages that cannot be successfully processed by FRONT ARENA.	If a message is not successfully converted to a business object, it is stored here.
WCS_Parallel Inbound	The adapter does not make use of a parallel inbound queue.
	ICS_InProgress queue
	The original versions of valid messages sent from FRONT ARENA to InterChange Server are stored here until the adapter completes processing; when processing is completed, the original message is moved to the local Archive queue.
	ICS_Archive_queue
	When a message has been fully processed by the adapter and sent to InterChange Server from FRONT ARENA, the original version of the message is stored here.

To enable communication between the two systems, use channels and transmission queues.

Channels that perform roles seen in the following table must be created on each machine for this topology.

Table 9. Channel roles

Channels On the FRONT ARENA machine	Channels On the InterChange Server machine
Sender_WCS	Sender_ICS
Receiver_ICS	Receiver_WCS

Creating the channels: In the following example instructions, specific server and queue manager names are specified so that they correlate the different machines

and queues. Channels are used to make sure that the correct queues refer to each other; the `local` versions of the queues are used to hold the actual information.

FRONT ARENA machine configuration tasks:

Note: The channel names used here are examples only.

1. Create two channels in the FRONT ARENA system using WebSphere MQ Explorer. One sender channel named `WCS` and one receiver channel named `ICS`.
2. Create a local queue, for example, using the name `ToICSSystem`.
3. Set the `ToICSSystem` queue as the transmission queue.
4. Set the following properties for the `WCS_Outbound` queue.
   a. Remote queue name `ICS_InboundRemote` queue manager name `ICS_server_name.queue.manager`. For example, `ICS.queue.manager`.
   b. Set the transmission queue name property to `ToICSSystem` as created in step 2.
5. To configure the sender channel do the following:
   a. Specify the connection name with the IP address and the port, for example, 9.182.12.235(1414). Where, 9.182.12.235 is the IP address of the machine where InterChange Server is running and 1414 is the default listener port.
   b. Specify the transmission queue name as `ToICSSystem`.

This completes the configuration tasks for the FRONT ARENA machine.

InterChange Server machine configuration tasks:

1. Create two channels using WebSphere MQ Explorer: One sender channel named `ICS` and one receiver channel named `WCS`.

   Note: The name of the sender channel in the WebSphere business integration system must be identical to the name of the receiver channel in FRONT ARENA. The name of the receiver channel in the WebSphere business integration system must be identical to the name of the sender channel in FRONT ARENA.
2. Create a new local queue, for example `ToICSSystem`. Set the `ToICSSystem` queue as the transmission queue.
3. Create a remote definition queue in the WebSphere business integration system. This remote definition queue must be used in the connector component as the output queue. Set the following properties:
   a. Remote queue name `WCS_SerialInbound`
   b. Remote queue manager name `<wcssytems_Q_manager_name>`. For example, QM_wcsfvt3.
   c. Set the transmission queue name property to `ToICSSystem`.
4. To configure the sender channel do the following:
   a. Specify the connection name with the IP address and the port, for example, 9.182.12.18(1414). Where, 9.182.12.18 is the IP address of the machine where FRONT ARENA is running and 1414 is the default listener port.
   b. Specify the transmission queue name as `ToICSSystem`.
5. Once the WebSphere MQ queues and channels on both the FRONT ARENA machine and the InterChange Server machine are configured:
   a. Start the receiver channel.
   b. Next, start the sender channel.

This completes the configuration tasks for the InterChange Server machine.

Two-machine, one-queue manager topology

In this topology, FRONT ARENA is installed on one machine and InterChange Server and the Adapter for FRONT ARENA are installed on another machine. Only one instance of WebSphere MQ is running, and the queues used by both machines are managed by a single queue manager. This scenario uses only local queues.

This topology requires queues that perform the following roles:

- Inbound queue

  FRONT ARENA requires that this queue exist. However, the adapter does not make use of this queue in this solution.
- Serial inbound queue

  This queue is for sending messages from InterChange Server to FRONT ARENA.
- Outbound queue

  This queue is for receiving messages sent from FRONT ARENA to InterChange Server.
- In-progress queue

  The original versions of valid messages sent from FRONT ARENA to InterChange Server are stored here until the adapter completes processing; when processing is completed, the original message is moved to the local Archive queue.
- `Archive_Queue`

  When a message has been fully processed by the adapter and sent to InterChange Server from FRONT ARENA, the original version of the message is stored here.
- `Unsubscribed_Queue`

  If a message is successfully parsed but does not correspond to any business object supported by the adapter, it is stored here.
- FRONT ARENA error queue
- ICS_Error_Queue

  If a message is not successfully converted to a business object and sent to the InterChange Server, it is stored here.
- WCS_Error_Queue

  This queue stores messages that cannot be successfully processed by FRONT ARENA.
- ReplyTo Queue

  This queue is used only in configurations that are set up for synchronous data exchange.

Configuring WebSphere Application Server JMS settings

To configure the WebSphere Application Server to create the Java Messaging Service Connection Factory and JMS queues to work with WebSphere MQ, perform the following steps:

1. From the command prompt:
   a. Update your classpath variable by typing the following command on one line:

```
set classpath= %classpath%;
MQ_install_path\java\lib\com.ibm.mqjms.jar;
MQ_install_path\java\lib\com.ibm.mq.jar;
MQ_install_path\java\lib\com.ibm.mq.iopp.jar;
MQ_install_path\java\lib\com.ibm.ibmorb.jar;WAS_install_path\lib\ns.jar
```

MQ_install_path — The path in which you installed WebSphere MQ

WAS_install_path — The path in which you installed the WebSphere Application Server

b. Add a new environment variable named MQ_JAVA_INSTALL_PATH by typing the following command:

```
set MQ_JAVA_INSTALL_PATH=MQ_install_path\java
```

MQ_install_path — The path in which you installed WebSphere MQ

c. Update the environment to use the JDK (Java Development Kit) that comes with WebSphere Application Server by typing the following command:

```
set PATH = WAS_Intall_Path\Java\bin;%PATH%
```

WAS_install_path — The path in which you installed the WebSphere Application Server

2. Ensure that the WebSphere Application Server is running and the correct classpath and environment variables defined in Step 1 above are added. Also check to make sure the JDK being used is the one in WebSphere Application Server by executing `java -version` and checking the version with the one found in *WAS_Install_Path*`\Java\bin`.
3. In the `MQ_install_path\java\bin directory`, open the `JMSAdmin.config` file and set the following values:

   ```
   INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory
   PROVIDER_URL=iiop://localhost:900
   ```

   The values above assume that FRONT ARENA and WebSphere MQ are installed on the same machine.

   ```
   SECURITY_AUTHENTICATION=none
   ```

4. Run the `MJSAdmin` program by providing the `JMSAdmin.config` file as a command line input:

   CommandPrompt:> `JMSAdmin -cfg JMSAdmin.config -t -v`

   Executing this command should enable you to lookup the JNDI (Java Naming and Directory Interface) service provided by WebSphere Application Server. You will see an `InitCtx>` prompt that you can use to run the JMS administration commands.
5. Register the QueueConnectionFactory and set the coded character set identifier by typing the following commands:
   - `define qcf (JMS_QueueConnection_Factory) qmanager (Your_QueueManager_Name)`
   - `alter qcf (JMS_QueueConnection_Factory) ccsid(1208)`

     Where `JMS_QueueConnection_Factory` is the name of the MQQueueConnectionFactory JMS object.

   When you execute the above set of commands, an entry for this queue connection factory is created in the WebSphere Application Server database under the BINDINGBEANTBL table. These objects are registered in the WebSphere Application Server database.

6. Define the following JMSQueues to map to the names you have established for your WebSphere MQ queues and the WebSphere MQ queue manager you are using. You can customize the JMSQueues names according to your own requirements, but the WebSphere MQ names to which you define them must match exactly, including casing, queue names you have established in WebSphere MQ. If you are using the batch file provided with this adapter for creating the appropriate WebSphere MQ queues for a single-machine topology, be sure to use those generated WebSphere MQ queue names as the values to which you define the JMS queues, as described in the table below.

   The following syntax defines the JMS Queues:

Note: If you are using a remote queue definition for the outbound queue, as you would in a two-machine, two-queue manager topology, the JMS_Outbound_Queue should not be defined to a local WebSphere MQ queue. If you are using a remote queue definition, the syntax for the outbound queue would be: `define q(JMS_Outbound_Queue)qmanager(Your_Queue_Manager_Name)`

```
define q(JMS_Outbound_Queue)qmanager
 (Your_Queue_Manager_Name)
 queue (Your_Outbound_QueueName)

define q(JMS_Inbound_Queue)qmanager
 (Your_Queue_Manager_Name)
 queue (Your_Inbound_QueueName)

define q(JMS_Parallel_Inbound_Queue)qmanager
  (Your_Queue_Manager_Name)queue
    (Your_Parallel_Inbound_Queue_Name)

define q(JMS_Serial_Inbound_Queue)qmanager
  (Your_Queue_Manager_Name)queue
   (Your_Serial_Inbound_Queue_Name)

define q(JMS_Error_Queue)qmanager
  (Your_Queue_Manager_Name)
   queue (Your_Error_Queue_Name)
```

Table 10. Defining JMS queue names

Your_Outbound_QueueName	The WebSphere MQ queue created for the outbound queue. By default, this is the queue which the adapter polls to pick up messages from FRONT ARENA to pass to InterChange Server. In the default WebSphere MQ queue setup created by the batch file, this value should be WCS_Outbound.
Your_Serial_Inbound_Queue	The WebSphere MQ queue created for the serial inbound queue. This is the queue into which the Adapter for FRONT ARENA places messages sent from InterChange Server to FRONT ARENA. In the default WebSphere MQ queue setup created by the batch file, this value should be WCS_Serial_Inbound
Your_Parallel_Inbound_Queue_Name	This is the WebSphere MQ queue created for the parallel inbound queue.
Your_Error_Queue_Name	The WebSphere MQ queue created for the error queue. This is where the Adapter for FRONT ARENA sends messages when it encounters an error with the message. In the default WebSphere MQ queue setup created by the batch file, this value should be WCS_Error.

Table 10. Defining JMS queue names (continued)

Your_Queue_Manager_Name	The name of the queue manager handling WebSphere MQ queues in your set up for the FRONT ARENA system. In a typical single machine setup, such as the setup created by the batch file ConfigureAdapterQueues.bat, the queue manager that you established for ICS would be used to manage the queues for the FRONT ARENA system as well. For such a setup, the default would be <*InterchangeServerName*>.queue.manager.

After creating the queues, set the following property for the outbound and error queues using the JMSAdmin console. This procedure specifies that JMS is dealing with a native WebSphere MQ application.

- alter q(JMSOutboundQueue) targclient(MQ)
- alter q(JMSErrorQueue) targclient(MQ)

Type end to exit the JMSAdmin tool. This completes your task for configuring the Java Messaging Service with WebSphere Application Server running FRONT ARENA.

Configuring the JMS ConnectionSpec within FRONT ARENA

Note: The JMSAdmin names and JMS connection factory must be the same as the values entered in the connectionSpec section of Commerce Configuration Manager, in the instance XML file. You can find the details under the Transports section in the FRONT ARENA Configuration Manager. Also see the instructions below.

Start the FRONT ARENA Administration Console. Log in as a Site Administrator, go to the Configuration section and choose the Transport option. Select WebSphere MQ as your transport and change the status to active. Log out from the Administration Console.

The FRONT ARENA solution requires the creation and use of a "store," as described in the *FRONT ARENA Installation Guide.* When you have completed publishing the store as described in "Publishing a Sample Store" section of that guide, log into the Administration Console, this time as a Store Administrator, and select the store you are using. In the Configuration section, add MQ Transport to the store. An entry for this is made in the STORETRANS table.

To enable the messaging system transport adapter, launch FRONT ARENA Configuration Manager and do the following:

1. Select Host name ->Instance.
2. Open the Components folder.
3. Select, TransportAdapter.
4. Select the Enable Component check box.
5. Click Apply.

Configure JMSQueue names and JMS Connection Factory with the values that you are using for the connectionSpec in this instance, as follows:

1. Select the Host name ->Instance.
2. Select Transports.

3. Expand `Outbound->JMS`.
4. Select the ConnectionSpec.
5. Input the ConnectionFactory name created when configuring the JMS settings for WebSphere Application Server.
6. Enter the Inbound, Error and Outbound queue names created above.
7. Click Apply.
8. Expand `Inbound->JMSInbound CCF Connector-Serial`.
9. Select the ConnectionSpec.
10. Input the ConnectionFactory Name, SerialInbound, Error and Output JMS queues.
11. Click `Apply`.
12. Expand `Inbound-JMSInbound CCF Connector-Parallel`.
13. Select the ConnectionSpec.
14. Input the ConnectionFactory, ParallelInbound, Error and Output JMS queues.
15. Click Apply.

Exit the Configuration Manager.

Updating FRONT ARENA JVM settings

You must update the WebSphere Application Server class path for the instance, adding the additional jar file entries. To do so, open the WebSphere Application Server Advanced Administrative Console and complete the following:

1. Select the host on which you are running your FRONT ARENA instance.
2. Select the WebSphere Administrative Domain.
3. Select `Nodes`.
4. Select your host name.
5. Select `Application Servers`.
6. Select the FRONT ARENA Server instance_name, where instance_name is the name of your FRONT ARENA instance.
7. Go to the JVM settings of the instance.
8. Select, `Add a new system property`.
9. Type in the following system property:

   ```
   name= ws.ext.dirs value=MQ_INSTALL_PATH/java/lib
   ```

   Where MQ_INSTALL_PATH is the path where you installed WebSphere WebSphere MQ.
10. Restart the WebSphere Application Server service for all the changes to take affect.

Enabling WebSphere MQ for the adapter

The Adapter for FRONT ARENA is a customized WebSphere MQ technology adapter. An instance of the WebSphere MQ adapter is assumed to be installed as the base of the Adapter for FRONT ARENA. This includes the definition of all WebSphere MQ queues used by the adapter.

For details on how to install a WebSphere MQ adapter see the corresponding documentation which is part of the WebSphere Interchange Server delivery. It is recommended to give the adapter a descriptive name, for example, FrontArenaConnector.

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

Default adapter configuration properties

As a variant of the WebSphere MQ technology adapter, the adapter for FRONT ARENA has the same connector configuration properties as its base adapter. Only the properties with a particular relevance for the MQ adapter for FRONT ARENA are presented below.

Table 11. Standard properties

Property Name	Description	Default value
ApplicationName	The connector's name.	FrontArenaConnector
MessageFileName	The message file used by the connector.	FrontArenaConnector.txt (This is the name for the message file that is shipped with the adapter.)

Table 12. Connector-specific properties

Property Name	Description	Default value
DataHandlerConfigMO	The meta-object used by the data handler.	MO_DataHandler_FrontArena Config (This data handler meta-object is shipped with the adapter.)
DataHandlerMimeType	The data handler selection based on MIME type.	FA/native
ConfigurationMetaObject	The default configuration meta-object.	MO_FrontArena_DefaultConfig (This configuration meta-object is shipped with the adapter.)

Table 13. Supported business objects

Business object name	Description
MO_DataHandler_FrontArenaDefaultConfig	The default data handler meta-object.
MO_DataHandler_FrontArenaNativeConfig	The data handler meta-object for native FRONT ARENA message format.
MO_FrontArena_DefaultConfig	The default configuration meta-object.
FA_trade	The example application-specific business object.

Note: For all supported business objects, agent support must be turned on.

Verifying the installation

The sections below describe the paths and filenames of the product after installation.

Note: WebSphere MQ and JMS typically are installed in separate directories in both Windows and UNIX environments. On an AIX system for example,

WebSphere MQ is installed by default in /var/mqm/ while JMS is installed in /usr/mqm/java/lib. You may want to redirect the JMS install to /var/mqm/java/lib to avoid deletion by routine /usr-related system administration tasks. Likewise on Windows, WebSphere MQ is generally installed under \Program Files\WebSphere MQ and JMS under \Program Files\IBM\MQSeries\Java. Update your classpath accordingly in the WebSphere MQ connector startup script.

Windows file structure

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

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

The table below 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.

Subdirectory of *ProductDir*	Description
WICS\connectors\WebSphereMQ\CWWebSphereMQ.jar	Contains classes used by the WebSphere MQ connector only.
WICS\connectors\WebSphereMQConnector\start_WebSphereMQ.bat	The startup script for the connector (NT/2000).
WICS\connectors\messages\WebSphereMQConnector.txt	The message file for the connector.
WICS\repository\WebSphereMQ\CN_WebSphereMQ.txt	The repository definition for the connector.
WICS\connectors\WebSphereMQ\samples\LegacyContact\WebSphereMQConnector.cfg	The example WebSphere MQ configuration file.
WICS\connectors\WebSphereMQ\samples\LegacyContact\PortConnector.cfg	The example Port connector configuration file.
WICS\connectors\WebSphereMQ\samples\LegacyContact\Sample_WebSphereMQ _LegacyContact.xsd	The example schema.
WICS\connectors\WebSphereMQ\samples\LegacyContact\Sample_WebSphereMQ _MO_Config.xsd	The example meta-object.
WICS\connectors\WebSphereMQ\samples\LegacyContact\Sample_WebSphereMQ _MO_DataHandler.xsd	The example data handler meta-object.
WICS\connectors\WebSphereMQ\samples\LegacyContact\Sample_WebSphereMQ _MO_DataHandler_DelimitedConfig.xsd	The example delimited data handler meta-object.
WICS\connectors\WebSphereMQ\samples\LegacyContact\Sample_WebSphereMQ _DynMO_Config.xsd	The example dynamic meta-object.
WICS\connectors\WebSphereMQ\samples\LegacyContact\JMSPropertyPairs.xsd	The example JMS properties.
WICS\DataHandlers\FrontArenaDataHandler.jar	The data handler.
WICS\repository\FrontArena\MO_DataHandler_FrontArenaDefaultConfig.xsd	The data handler meta-objects.
WICS\repository\FrontArena\MO_DataHandler_FrontArenaNativeConfig.xsd	The data handler meta-objects.
WICS\repository\FrontArena\MO_FrontArenaDefaultConfig.xsd	The adapter configuration meta-object.
WICS\repository\FA_trade.xsd	The example application-specific business object.
WICS\repository\FA_instrument.xsd	The example application-specific business object.
WICS\bin\Data\App\FrontArenaConnectorTemplate	The configuration template.

Table 14. Bridge for FRONT ARENA files

Subdirectory of *ProductDir*	Description
WICS\other\FrontArena\bia_bfa.exe	The bridge executable file.
WICS\other\FrontArena\bia_bfaMessages.txt	The bridge messages file.
WICS\other\FrontArena\bia_BFAAlert.jar	The bridge jar file for SMTP e-mail alert mechanism.
WICS\other\FrontArena\config.sample	The example bridge configuration file.

Note: All product path names are relative to the directory where the product is installed on your system. In the case of the bridge, most likely, it will be installed on a separate machine.

UNIX file structure

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

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

The table below 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.

Subdirectory of *ProductDir*	Description
connectors/WebSphereMQ/CWWebSphere MQ.jar	Contains classes used by the WebSphere MQ connector only.
connectors/WebSphereMQ/start_WebSphereMQ.sh	The 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/messages/WebSphereMQ/Connector.txt	The message file for the connector.
repository/WebSphereMQ/CN_WebSphere MQ.txt	The repository definition for the connector.
connectors/WebSphereMQ/samples/LegacyContact/WebSphereMQConnector.cfg	The example WebSphere MQ configuration file.
connectors/WebSphereMQ/samples/LegacyContact/PortConnector.cfg	The example port connector configuration file.
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _LegacyContact.xsd	The example schema.
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_Config.xsd	The example meta-object.
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_DataHandler.xsd	The example data handler meta-object.
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _MO_DataHandler_DelimitedConfig.xsd	The example delimited data handler meta-object.
connectors/WebSphereMQ/samples/LegacyContact/Sample_WebSphereMQ _DynMO_Config.xsd	The example dynamic meta-object.
connectors/WebSphereMQ/samples/LegacyContact/JMSPropertyPairs.xsd	The example JMS properties.
WICS/DataHandlers/FrontArenaDataHandler.jar	The data handler.
WICS/repository/FrontArena/MO_DataHandler_FrontArenaDefaultConfig.xsd	The data handler meta-objects.
WICS/repository/FrontArena/MO_FrontArena_DefaultConfig.xsd	The adapter configuration meta-object.
WICS/repository/FA_trade.xsd	An example application-specific business object.
WICS/bin/Data/App/FrontArenaConnectorTemplate	The connector template.
WICS/other/FrontArena/bia_bfa	The bridge executable file.
WICS/other/FrontArena/bia_bfaMessages.txt	The bridge messages file.
WICS/other/FrontArena/bia_BFAAlert.jar	The bridge jar file for SMTP e-mail alert mechanism.
WICS/other/FrontArena/config.sample	The example bridge configuration file.
WICS/other/FrontArena/start_bfa.sh	The bridge start script.

Table 15. Bridge for FRONT ARENA files

Subdirectory of *ProductDir*	Description
WICS/other/FrontArena/bia_bfa	The bridge executable file.
WICS/other/FrontArena/bia_bfaMessages.txt	The bridge messages file.
WICS/other/FrontArena/bia_BFAAlert.jar	The bridge jar file for SMTP e-mail alert mechanism.

Table 15. Bridge for FRONT ARENA files (continued)

Subdirectory of *ProductDir*	**Description**
WICS/other/FrontArena/config.sample	A example bridge configuration file.
WICS/other/FrontArena/start_bfa.sh	The bridge start script.

Note: All product path names are relative to the directory where the product is installed on your system. In the case of the bridge, it will most likely be installed on a separate machine.

Chapter 3. Business objects for FRONT ARENA

This chapter describes how the adapter processes business objects and describes the assumptions that the connector makes. You can use this information as a guide to implementing new business objects.

The adapter comes with example business objects only. You must build your specific business objects. The business object definitions must match the FRONT ARENA business objects stored in the FRONT ARENA database. Business object definitions can be created manually using the Business Object Designer, or automatically by using the JDBCODA.

Business object metadata

The adapter is metadata-driven. In this context, metadata is data about the application, which is stored in a business object definition and helps the adapter 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 that are 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 text. Because the connector is metadata-driven, it can handle new or modified business objects without requiring modifications to the adapter code. The adapter's configured data handler makes assumptions about the structure of its business objects, object cardinality, the format of the application-specific text, and the database representation of the business object. Therefore, when you create or modify a business object for the adapter, your modifications must conform to the rules that the adapter is designed to follow, or it cannot process new or modified business objects correctly.

Business object structure

The adapter for FRONT ARENA retrieves WebSphere MQ messages from a queue and attempts to populate business objects (defined by the meta-object) with the business data contained in the message.

The business data in the MQ Series messages exchanged by the adapter is contained within XML documents. The adapter uses the FRONT ARENA data handler to convert the data from FRONT ARENA native messages into business objects, and from business objects into FRONT ARENA native messages.

The structure of the business object definitions must conform to the requirements of the FRONT ARENA business objects. For information about how to use the JDBCODA or generate business object definitions from the FRONT ARENA database, refer to the *Installation Guide for WebSphere Business Integration Adapters* and the *Adapter for JDBC User Guide*.

To see the properties and structure used in a typical Trade business object definition for FRONT ARENA, open the file, /DataHandlers/repository/FrontArena/FA_trade.xsd.

Custom business object handler

A static meta-object is used to provide the adapter with some basic conversion properties.

Table 16. Static configuration meta-object, MO_FrontArena_DefaultConfig:

Property name	Application specific information
Default	OutputQueue=queue://DENALI/FRONTARENA.OUT?targetClient=1;ResponseTimeout=60000;
FA_trade_Create	InputFormat=INSERT;
FA_trade_Update	InputFormat=UPDATE;
FA_trade_Delete	InputFormat=DELETE;
FA_trade_Retrieve	InputFormat=SELECT;

The static meta-object consists of a list of conversion properties defined for different business objects. To define the conversion properties for a business object, create a string attribute and name it using the syntax <business object name>_<verb>. For example, to define the conversion properties for an `FA_trade` object with the verb `Create`, create an attribute named `FA_trade_Create`.

In the application-specific information of the attribute, you can specify the actual conversion properties. Additionally, a reserved attribute named `Default` can be defined in the meta-object. When this attribute is present, its properties act as default values for all business object conversion properties.

Seen above, the example meta-object defines the conversion properties for the business object `FA_trade` and all supported verbs. For additional business object and verb combinations to be supported, the appropriate entries must be added.

The `Default` property contains the name of the WebSphere MQ queue, `OutputQueue`, to be used for sending service requests to the bridge component of the adapter. The queue name must be adapted to the customer's setup.

The queue name,`targetClient=1`, must be appended to the queue name in any case, however. This ensures that the messages sent to the bridge component of the adapter have the format expected by the FRONT ARENA application.

Note: The bridge component of the adapter expects native WebSphere MQ messages consisting of an MQMD header and the message body only. If this parameter is not supplied, JMS style messages are created by the adapter. They contain an additional `MQRFH2` header.

The parameter `ResponseTimeout` indicates the period of time, in milliseconds, to wait before timing out when waiting for the response to a service request. The connector returns `SUCCESS` immediately without waiting for a response if this is left undefined or with a value less than zero. This parameter must be set if service requests are to be processed in a synchronous fashion.

For each additional business object and verb combination to be supported, a corresponding new property must be added. Only the parameter `InputFormat` must

be specified. This parameter is used to map the FRONT ARENA verbs `INSERT`, `UPDATE`, `DELETE`, and `SELECT`. These verbs are stored in the `FORMAT` field of the messages received from the bridge to the WBI verbs `Create`, `Update`, `Delete`, and `Select`.

For more details on these parameters, see the documentation for the *Adapter for WebSphere MQ User Guide*.

Chapter 4. Configuring the data handler

This chapter describes how to configure the data handler to processes and convert messages.

FRONT ARENA messages and the data handler

The FRONT ARENA data handler is required for using the adapter with the FRONT ARENA application. There are two ways to configure the data handler:

- Specify the data handler class name in the connector-specific property `DataHandlerClassName`. For more information, see "Connector-specific properties" on page 74.
- Specify the mime type and the data handler meta-object that defines the configuration for that mime type in the connector-specific properties DataHandlerMimeType and DataHandlerConfigMO, respectively. For more information, see Table 38 and the *Data Handler Guide*.

To understand the data handler's way of processing, it is essential to comprehend the structure of FRONT ARENA messages. These messages have the following characteristics:

- A message is a list containing basic elements or nested lists.
- The beginning and the end of a list are indicated by the message elements `<list>` and `</list>` where `<list>` is the list's name. List names are arbitrary and they do not have to be unique.
- Simple list elements have a keyword = value format.
- The name of the top level list is always MESSAGE. It has five elements:
  - TYPE=<name of the created/modified/deleted/retrieved object and operation applied to it>
  - VERSION=<FRONT ARENA AMB version>
  - TIME=<message creation timestamp>
  - SOURCE=<name of the message originator>
  - List representing the created or modified object
- Subordinated lists must be at the end of their parent list. No simple list elements may follow them. There may be several child lists, however.

The entire message is in text format. Independent of their logical format, all values of simple elements are represented as strings. The list tags and the keywords are always in upper case.

The following incomplete message indicates the creation of a new trade:

```
[MESSAGE]
 TYPE=INSERT_TRADE
 VERSION=1.0
 TIME=2003-09-15 18:21:36
 SOURCE=ATLAS
 [TRADE]
  STATUS=FO confirmed
```

```
 TRADER_USERNBR.USERID=INTAS
 QUANTITY=200
 PRICE=89.57
 TIME=2003-09-15 12:05:41
  [INSTRUMENT]
   ISIN=FR123456780
   EXPIRY_DATE=2001-05-12
  [/INSTRUMENT]
 [/TRADE]
[/MESSAGE]
```

In addition to the primary object and list TRADE, this message contains a nested list named INSTRUMENT.

Converting a FRONT ARENA message to a business object

For a given FRONT ARENA message, the name handler composes the name of the associated business object from the following sources: The value of the BOPrefix attribute of the data handler's meta-object, and the name of the (single) child list of MESSAGE, converted to lower case.

Note: Both name components are separated by an underscore.

For example, if you assume that the meta-object attribute BOPrefix has a value of FA, then, the business object, FA_trade, would be created from the message above.

The verb is derived from the content of the message's FORMAT field, according to the conversion properties specified in the static configuration meta-object.

Note: A static meta-object is used to provide the adapter with some basic conversion properties.

For each element of a list in the FRONT ARENA message, the data handler tries to find a corresponding business object attribute. To associate a business object attribute with a list element, either the business object attribute name and the element name (keyword for simple attributes) must be identical or the application specific information of a business object attribute must identify the FRONT ARENA message element corresponding to the attribute. Such an explicit mapping is made by specifying CN=<FRONTARENA element name> in the application specific information of a business object attribute.

Note: The <FRONT ARENA element name> is case insensitive. The CN must be in upper case.

Business object attributes corresponding to nested lists are expected to be child business objects. Otherwise, an error is reported. This means that the list hierarchy of the FRONT ARENA message must be reflected in the business object definition.

The structures of the FRONT ARENA messages and the associated business object definitions need not be completely identical. Elements and sections of the FRONT ARENA message that are not reflected in the business object definition are simply skipped. This applies to both simple attributes and (trees of) subordinate lists. For each message element skipped, an informational entry is added to the adapter's log file.

Business object attributes not represented in the original message are set to a constant value, CxIgnore.

Converting a business object to a FRONT ARENA message

Before actually processing the business object, the top-level MESSAGE list (that is not reflected in the business object) and its simple elements are created as follows:

- The TYPE element is composed of the value of parameter, `FAVERB`, of the business object verb's application specific information and the value of parameter, `TN`, of the business object's application specific information, separated by an underscore. The value of TYPE is in upper cases.
- VERSION is set to the constant value, 1.0.
- TIME is set to the current date and time.
- SOURCE is set to the value of attribute `SourceName` of the data handler meta-object.

The business object is processed in sequential order. For each simple attribute, a `keyword=value` pair is appended consisting of the value of parameter `CN` of the attribute application specific information or the attribute's name if the parameter is not specified, and the attribute's value.

If a child business object is encountered, a child list is opened. The list's name is the value of parameter `CN` of the attribute application specific info or the attribute's name if the parameter isn't specified. When all attributes of a business object have been processed, the closing tag of the associated list is added to the message.

Message business object example

A business object for the example message above might look as follows:

Table 17. Business object FA_trade

General object level application-specific information	**TN=trade**
Verb name	Application-specific information
Create	FAVERB=insert
Delete	FAVERB=delete
Retrieve	FAVERB=select
Update	FAVERB=update
Attribute name	Application-specific information
trader	CN=trader_usrnbr.userid
quantity	CN=quantity
price	CN=price
instrument (child business object)	CN=instrument

Table 18. Child business object FA_instrument

General object level application-specific information	**TN=trade**
Verb name	Application-specific information
Create	FAVERB=insert
Delete	FAVERB=delete
Retrieve	FAVERB=select
Update	FAVERB=update
Attribute name	Application-specific information
isin	
expirationdate	FAVERB=insert

The business object definition does not have any attributes corresponding to the TRADE elements, STATUS and TIME. As mentioned before, the business object does not have to contain all the attributes of the corresponding FRONT ARENA message.

For the attribute, isin, of the child business object FA_instrument, no mapping is needed in the application specific information because the attribute name and the name of the corresponding FRONT ARENA message element are identical.

Data handler meta-object configuration

The data handler is configured by means of two meta-objects:

- MO_DataHandler_FrontArenaDefaultConfig
- MO_DataHandler_FrontArenaNativeConfig

Table 19. Data handler meta-object, MO_DataHandler_FrontArenaDefaultConfig

Property name	Description	Possible values	Default value
FA_native	The generic top-level data handler meta-object, which currently only supports MIME type FA/native.	MO_DataHandler_FrontArenaNativeConfig (points to the child meta-object described below).	N/A

Table 20. Data handler meta-object, MO_DataHandler_FrontArenaNativeConfig (for MIME type FA/native)

Property name	Description	Possible values	Default value
ClassName	The class containing the data handler implementation.	com.ibm.adapters.datahandlers.frontarenaFANativeDataHandler	com.ibm. adapters. datahandlers.frontarena. FANative Data Handler
NameHandler Class	The class containing the name handler implementation.	com.ibm.adapters.datahandlers.frontarena.FANative NameHandler	com. ibm. adapters. datahandlers.frontarena. FANativeNameHandler
BOPrefix	The delimiter separating, and terminating, elements of FRONT ARENA messages.	Any sequence of characters.	FA
Attribute Delimiter	The keyword-value separator used in simple elements of FRONT ARENA messages.	The sequence of characters separating elements in FRONT ARENA messages. **Note**: FRONT ARENA versions 1.x and 2.0 use the newline character as a separator. The value of this property only has to be changed if future FRONT ARENA versions use a different separator.	\n (newline)

Table 20. Data handler meta-object, MO_DataHandler_FrontArenaNativeConfig (for MIME type FA/native) (continued)

Property name	Description	Possible values	Default value
KeyValue Separator	The value of the simple list element, SOURCE, of the top level list, MESSAGE.	The separator used between the keyword and the value of simple elements of FRONT ARENA messages. **Note**: FRONT ARENA versions 1.x and 2.0 use the equal sign character as a separator. The value of this property must be changed only if future FRONT ARENA versions use a different separator.	=
SourceName	Default verb	Any sequence of characters.	WBI
DefaultVerb	Default verb	Create, Update, Delete, or Retrieve.	Create
CxIgnore	When converting from a business object, the data handler assigns the value of this attribute to a list element when it encounters a business object with the CxIgnore constant as its value. When converting to a business object, the data handler assigns the CxIgnore constant to a business object attribute when the associated message element has the value of this attribute as its value.	Any sequence of characters.	CxIgnore
SkipCxIgnore	When this attribute is set to true, the data handler ignores all business object attributes / list elements with a value of CxIgnore.	True or false.	False

Sending requests without notification

To configure the adapter to send requests without notification (the default asynchronous mode, also known as "fire and forget"):

- Create a business object that represents the request you want to send and is compatible with the data handler.

- Use either a static or a dynamic meta-object to specify the target queue and format. For more on static and dynamic meta-objects, see “Meta-objects configuration” on page 83 and “Overview of creating dynamic child meta-objects” on page 90..
- Set the property `ResponseTimeout` in the (static or dynamic) meta-object to `-1`. This forces the connector to issue the business object without checking for a return.

Data handler event notification support

If the FRONT ARENA application is set up accordingly, it publishes a message by way of its AMB middleware each time a new object is created or an existing one is modified or deleted. The bridge component of the adapter subscribes to all these messages. Each event message received is forwarded to a WebSphere MQ queue.

In order to enable the adapter to handle these events, it must be configured to use this queue as its `InputQueue` (in its connector specific properties).

Note: To make sure that no events are lost, this queue should be set up as a persistent queue.

Chapter 5. Configuring the connector

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 42).

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 43 to set up a new one.

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

Starting Connector Configurator

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

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

Running Configurator in stand-alone mode

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

To do so:

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

You may choose to run Connector Configurator independently to generate the file, and then connect to System Manager to save it in a System Manager project (see "Completing a configuration file" on page 47.)

Running Configurator from System Manager

You can run Connector Configurator from System Manager.

To run Connector Configurator:

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

To edit an existing configuration file:

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

Creating a connector-specific property template

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

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

- To create a new template, see "Creating a new template" on page 43.
- To use an existing file, simply modify an existing template and save it under the new name. You can find existing templates in your `\WebSphereAdapters\bin\Data\App` directory.

Creating a new template

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

To create a template in Connector Configurator:

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

Specifying general characteristics

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

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

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

Specifying values

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

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

To create a new property value:

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

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

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

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

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

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

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

Setting dependencies

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

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

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

   == (equal to)

   != (not equal to)

   > (greater than)

   < (less than)

   >= (greater than or equal to)

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

Creating a new configuration file

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

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

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

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

Creating a configuration file from a connector-specific template

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

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

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

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

- **System Connectivity**

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

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

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

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

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

Using an existing file

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

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

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

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

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

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

- Configuration (*.cfg)
- ICS Repository (*.in, *.out)

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

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

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

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

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

Completing a configuration file

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

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

1. Under the **Standard Properties** tab, select the value field for the BrokerType property. In the drop-down menu, select the value ICS, WMQI, or WAS.
2. The Standard Properties tab will display the properties associated with the selected broker. You can save the file now or complete the remaining configuration fields, as described in "Specifying supported business object definitions" on page 50..
3. When you have finished your configuration, click **File>Save>To Project** or **File>Save>To File**.

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

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

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

Setting the configuration file properties

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

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

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

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

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

- Associated Maps
- Resources
- Messaging (where applicable)

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

Standard properties differ from connector-specific properties as follows:

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

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

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

Setting standard connector properties

To change the value of a standard property:

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

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

Setting application-specific configuration properties

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

1. Right-click in the top left portion of the grid. A pop-up menu bar will appear. Click **Add** to add a property. To add a child property, right-click on the parent row number and click **Add child**.
2. Enter a value for the property or child property.
3. To encrypt a property, select the **Encrypt** box.
4. Choose to save or discard changes, as described for "Setting standard connector properties" on page 48.

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

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

Encryption for connector properties

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

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

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

Update method

Refer to the descriptions of update methods found in the *Standard configuration properties for connectors* appendix, under "Setting and updating property values" on page 107.

Specifying supported business object definitions

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

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

If ICS is your broker

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

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

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

To delete a business object from the supported list:

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

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

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

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

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

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

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

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

If a WebSphere Message Broker is your broker

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

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

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

If WAS is your broker

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

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

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

Associated maps (ICS only)

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

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

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

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

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

- **Business Object Name**

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

- **Associated Maps**

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

- **Explicit**

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

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

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

  To explicitly bind a map:

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

Resources (ICS)

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

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

Messaging (ICS)

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

Setting trace/log file values

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

To change the logging and tracing values:

1. Click the **Trace/Log Files** tab.

2. For either logging or tracing, you can choose to write messages to one or both of the following:
   - To console (STDOUT):
     Writes logging or tracing messages to the STDOUT display.

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

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

Data handlers

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

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

Saving your configuration file

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

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

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

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

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

Changing a configuration file

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

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

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

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

Completing the configuration

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

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

Using Connector Configurator in a globalized environment

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

Connector Configurator supports non-English characters in:

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

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

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

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

```
                                <Value>es_ES</Value>
                                <Value>pt_BR</Value>
                                <Value>en_US</Value>
                                <Value>en_GB</Value>
                        <DefaultValue>en_US</DefaultValue>
                </ValidValues>
</Property>
```

Chapter 6. Configuring the bridge

This chapter describes how to set configuration property values for the bridge component of the adapter. The bridge between FRONT ARENA and WebSphere MQ reads and writes messages from and to an MQ queue in native FRONT ARENA message format. The messages on the queue can then be accessed by either the adapter for FRONT ARENA or other third party applications.

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.

Bridge parameters

The bridge executable component of the adapter for FRONT ARENA has a single parameter: The name and the location of the configuration file to be used.

This parameter is mandatory. If it is not supplied, the bridge displays a usage message and terminates immediately. In addition, if an invalid or inaccessible configuration file is specified, the bridge immediately terminates. An error message explaining the reason for the termination is displayed in the command window.

Configuration file

The bridge has a number of configuration options which are specified in a configuration file. This configuration file is divided in stanzas or sections.

Stanzas start with a name in square brackets, for example, `[stanza1]`, except for the first stanza, which does not need an explicit header. The first stanza ends with the beginning of a new stanza, or with the end of the configuration file.

Each stanza contains a set of logically related configuration parameters which are in `keyword=value` format. Keywords must be unique per stanza but the same keywords can be used in different stanzas.

Comments can be added to the configuration file at any point. Comment lines must start with a hash pragma (#) in the first column. Comment lines are simply ignored by the input parser. The same is true for blank lines which may be interspersed at any point.

Bridge configuration parameters

The configuration properties are read by the bridge at program start up time. Any changes applied to the configuration file are ignored until the bridge is stopped and re-started. If any of the mandatory stanzas or properties are missing, the adapter for FRONT ARENA terminates immediately.

Note: The stanza names, keywords, and values are case sensitive.

Configuration parameters, ordered by stanzas, are supported by the bridge.

General configuration settings

The main (unnamed) stanza contains some general configuration properties and the parameters used for communication with the AMB component of FRONT ARENA.

Table 21. Main (unnamed) stanza

Keyword	Mandatory/ Optional/ Default value	Possible values	Description
Mode	Mandatory	Event, Request, Both	The types of processing supported by the bridge include the following: • Event: Event processing only • Request: Request processing only • Both: Both event and request processing
FAHost	Mandatory	Valid URL	The URL of the system hosting FRONT ARENA.
FAAMBPort	Mandatory	1 - 65536	The port number of the FRONT ARENA AMB server.
FAUserID	Optional	1 - 65536 Valid FRONT ARENA (AMB) User ID	The user ID used when logging in to the AMB. **Note**: If no user ID and password are provided, the values from the `AMB.ini` file are taken.
FAPassword	Optional	Valid FRONT ARENA (AMB) password	The password used when logging in to the AMB.
FAMonitoring Port	Optional	1 - 65536	The port number of the FRONT ARENA System Monitor. **Note**: This is supported with FRONT ARENA 1.5 or higher only.
FARetries	Optional Default: 3	Integer value: Zero value means retry forever.	The number of retries if the connection to the AMB cannot be established or drops.
FARetryInterval	Optional Default: 60000	Integer value > 0	The time (in milliseconds) between successive retries to establish an AMB connection.

Table 21. Main (unnamed) stanza (continued)

Keyword	Mandatory/ Optional/ Default value	Possible values	Description
PollFrequency	Optional Default: 10000	Integer value > 0	The time interval (in milliseconds) between successive attempts to read messages from the AMB or WebSphere MQ queues. In the case of heavy traffic, the time interval is automatically reduced by the application.
TraceFile	Optional **Note**: This property must be provided if the trace is switched on (TraceLevel>0).	(Path and) file name	The trace file name.
TraceLevel	Optional Default: 0	0: Trace switched off 1: Error messages 2: Not used currently 3: Not used currently 4: Trace call sequence 5: Informational messages	The trace level. **Note**: Higher trace levels include all messages from lower levels.
LogFile	Mandatory	(Path and) file name	The log file name.
LogLevel	Optional Default: 3	1: Log errors only 2: Log errors and warnings 3: Log errors, warnings, and informational messages	The log level.

Event processing

The stanzas, seen in Table 22 on page 60, must be specified if the bridge is configured for event processing, for example, if the `Mode` parameter is set to `Event` or `Both`.

Stanza EventHandler mainly contains the WebSphere MQ parameters needed to establish a WebSphere MQ connection with a target application that is used to forward FRONT ARENA event notifications to the target application.

Table 22. Stanza EventHandler

Keyword	Mandatory/ Optional/ Default value	Possible values	Description
Name	Mandatory	The name of an existing AMB client.	Client name used for communication with FRONT ARENA. For more information, see Chapter 7, "Configuring the application," on page 71.
WMQQueueManager	Mandatory	The name of an existing WebSphere MQ queue manager.	WebSphere MQ queue manager used for communication with target application. For more information, see "Communicating with the WBI adapter" on page 66.
WMQChannel	Mandatory	<channel/ <transport>/ hostname(port)	WebSphere MQ channel used for communication with the given queue manager/ Transport (which usually is TCP) / URL of the system hosting the queue manager + queue manager listener port. For more information, see Chapter 7, "Configuring the application," on page 71.
WMQQueue	Mandatory	The name of an existing WebSphere MQ queue owned by the given queue manager.	The WebSphere MQ queue used to forwarding event messages to the target application. For more information, see Chapter 7, "Configuring the application," on page 71.
WMQUserID	Optional	A user ID which is known to the given queue manager.	The user ID to be used in messages sent by the bridge.
WMQRetries	Optional Default: 3	Integer value; Negative values mean: Retry forever.	The number of retries for the times that the WebSphere MW connection cannot be established or drops.
WMQRetry Interval	Optional Default: 60000	Integer value > 0	The time (in milliseconds) between successive retries to establish a WebSphere MQ connection.

Event subscription and filtering

FRONT ARENA publishes event notifications under topics. A topic consists of the following parts:

- A prefix that is defined by the FRONT ARENA AMBA configuration parameter followed by a slash (/). For more information on configuration parameters, see Chapter 7, "Configuring the application," on page 71.
- The type of object affected by the event.

Note: An update to a `TRADE` object would be published under the topic, `AMBA/TRADE` (assuming that `AMBA` is the prefix from 1).

A client application subscribing to this single topic would get notifications about any changes to `TRADE` objects but would not notice any modifications of `INSTRUMENT` objects. Instances of stanza EventHandler and Filter are primarily used to tell the

bridge component of the adapter which topics it has subscribed to. If the event filtering defined this way is too coarse-grained, additional filtering criteria can be specified in this stanza.

Each event notification message contains the following attributes:

- TYPE: Describes the operation that caused the event notification and the type of object it was applied to, in the following syntax:

  `<operation>_<object type>`
- SOURCE: Contains the originator of the event notification (which usually is the name of the FRONT ARENA AMBA)

In the example above, the TYPE would be `UPDATE_TRADE` and the SOURCE could be, for example, `ProdAmba`. The additional filter criteria which can be specified in this stanza use these two message attributes. They allow the bridge to be configured so that it only forwards event notifications caused by the update of a `TRADE` object and published by `ProdAmba`. In our example, this is accomplished with the following settings:

- Topics to `AMBA/TRADE`
- Verbs to `UPDATE`
- Sources to `ProdAmba`

Stanza EventHandler/Filter<n> (<n> being an integer value):

Table 23. Event notifications

Keyword	Mandatory/ Optional/ Default	Possible Values	Description
Topics	Mandatory	Any FRONT ARENA topic name (for example, EXT/TRADE).	FRONT ARENA topics (separated by semicolons) that the bridge will subscribe to.
Sources	Optional	Any FRONT ARENA source name (for example, AMBA).	Only messages from these sources (separated by semicolons) will be considered.
Objects	Optional	Any FRONT ARENA object (for example, TRADE).	These objects (separated by semicolons) shall be considered only. **Note**: This is the second token from the <operation>_<object type> pair described above.
Verbs	Optional	Any FRONT ARENA verb (which are INSERT, UPDATE, and DELETE).	Only these verbs (separated by semicolons) are considered. **Note**: This is the first token from the <operation>_<object type> pair described above.

Several instances of this stanza (with names EventHandler/Filter1, EventHandler2, and so on) can be specified.

As described above, each stanza defines a filter. The set of events subscribed to in the `Topics` property can be restricted by the following parameters: Sources, Objects, and Verbs. If several stanzas are provided, the set of events forwarded by the

bridge component of the adapter is the union of the sets specified by the different stanzas. All properties of this stanza may have multiple values, separated by semicolons.

Request processing

The following stanzas must be specified if the bridge component of the adapter is configured for request processing, for example, if the `Mode` parameter is set to `Request` or `Both`.

Whether requests are to be processed in a synchronous or asynchronous fashion is determined by either specifying or omitting the parameters `WMQReplyQueue` and `WMQCorrelationQueue`. To specify the stanzas, refer to the tables below.

Table 24. Request processing

Parameters WMQReplyQueue and WMQCorrelationQueue	Request Processing
None specified	Asynchronous
Both specified	Synchronous
One specified, second one not specified	Not allowed (configuration error)

Table 25. Stanza RequestHandler

Keyword	Mandatory/ Optional/ Default value	Possible values	Description
Name	Mandatory	The name of an existing AMB client.	The client name used for communication with the FRONT ARENA application. For more information, see Chapter 7, “Configuring the application,” on page 71.
WMQQueueManager	Mandatory	The name of an existing WebSphere MQ queue manager.	The WebSphere MQ queue manager used for communication with target application. For more information, see “Communicating with the WBI adapter” on page 66.
WMQChannel	Mandatory	<channel/ <transport>/ hostname(port)	The WebSphere MQ channel used for communication with the given queue manager/ Transport (which usually is TCP)/ URL of the system hosting the queue manager + queue manager listener port. For more information, see “Communicating with the WBI adapter” on page 66.
WMQRequestQueue	Mandatory	The name of an existing WebSphere MQ queue owned by the given queue manager.	The WebSphere MQ queue used to send requests to the bridge component of the adapter.

Table 25. Stanza RequestHandler (continued)

Keyword	Mandatory/ Optional/ Default value	Possible values	Description
WMQReplyQueue	Optional **Note**: This parameter must be specified if the WMQ Correlation Queue is specified.	The name of an existing WebSphere MQ queue owned by the given queue manager.	The WebSphere MQ queue used to pass replies back to the originator of the associated request.
WMQCorrelation Queue	Optional **Note**: This parameter must be specified if the WMQReply Queue is specified.	The name of an existing WebSphere MQ queue owned by the given queue manager.	The WebSphere MQ queue internally used by the bridge component of the adapter for request - reply correlation.
WMQUserID	Optional	A user ID that is known to the given queue manager.	The user ID to be used in the messages sent by the bridge component.
WMQRetries	Optional Default: 3	Integer value; Negative values mean: Retry forever.	The number of retries if the WebSphere MW connection cannot be established or drops.
WMQRetryInterval	Optional	Integer value > 0	The time (in milliseconds) between successive retries to establish a WebSphere MQ connection.
FAPublicationSubject	Mandatory	String	The topic under which request messages are published on the AMB by the bridge. For more information, see Chapter 7, "Configuring the application," on page 71.
FAAMBASenderSource	Optional	String	The prefix of the subject under which FRONT ARENA publishes replies to requests. For more information, see Chapter 7, "Configuring the application," on page 71.

Even if the parameters, for example `WMQQueueManager`, `WMQChannel`, or others, were specified in an EventHandler stanza already, they must be re-specified. The same or different values can be taken.

Note: In most scenarios, it is appropriate to use the same values.

User defined keys and FRONT ARENA keys

You can create objects in the FRONT ARENA system from a collaboration and update or delete such objects by defining a key for each object type and verb to be supported. This key is needed to correlate replies with the originating requests. The FRONT ARENA application does not provide a consistent internal mechanism for this.

FRONT ARENA keys

For each object created in the FRONT ARENA system, an internal key is generated by the FRONT ARENA application. This key is unique for each object type. For example, all `TRADE` objects have different internal keys. Internal keys are stored persistently in the primary database table of the object. The column used for storing the key depends on the object type.

Examples for the internal key for `TRADE` objects are stored in column `trdnbr` of the `TRADE` database table. Examples for the internal key for `INSTRUMENT` objects are stored in the column, `insaddr` of the `INSTRUMENT` database table.

AMB message IDs

Each message that is put on the FRONT ARENA AMB bus is assigned a unique message ID by the AMB. This message ID is made available to the AMB client application issuing the message.

If a message is rejected by FRONT ARENA, the error reply returned by the system contains the message ID of the causing message.

This is exploited by the bridge for correlating requests and error replies. The AMB message ID is not externalized by the bridge, however. It is neither part of the reply message sent to the adapter for FRONT ARENA nor part of the business object built by the adapter.

FRONT ARENA keys as user-defined keys

Obviously, the FRONT ARENA key cannot be used in create (insert) scenarios for request-reply correlation. In the cases of updates and deletions it can be used, provided this internal key is known to the requestor.

In scenarios that include all types of requests, the FRONT ARENA keys are not usually utilized. Relationship tables must be maintained and used by the collaborations that are involved for correlating FRONT ARENA objects with the same objects in other systems. Typically, the user-defined key will be used as a key for FRONT ARENA objects in these tables as well.

If different keys are used for different types of requests, several of these tables are required. Thus, the complexity of the collaborations will be increased significantly.

Candidates for user-defined keys

Obviously, only attributes of an object that can be assigned (or internally are assigned) unique values are candidates for user-defined keys. Furthermore, the corresponding database columns must be part of an index.

Candidates can be found in the Keys and Indexes chapter of the FRONT ARENA Data Model Reference (FCA 1051).

Examples:

- For `TRADE` objects, optional_key can be used in any case.
- For update and delete requests, as explained above, trdnbr can be used.
- For `INSTRUMENT` objects, isin, extern_id1, and extern_id2, can be used in any case.
- For update and delete requests, `insaddr` and `insid` can be used.

Table 26. Stanza RequestHandler/ObjectKeys:

Keyword	Mandatory/ Optional/ Default value	Possible values	Description
<requesttype>_<object>	Mandatory	The name of the attribute of <object>.	The name of the attribute of <object> to be used as the user key if a request of type <request type> is performed for <object>.

Valid <request type>'s: `INSERT`, `UPDATE`, and `DELETE`.

Valid <object>'s: `TRADE`, `INSTRUMENT`, etc. (any FRONT ARENA objects)

Example:

```
[RequestHandler/ObjectKeys]
INSERT_TRADE = OPTIONAL_KEY
UPDATE_TRADE = TRDNBR
DELETE_TRADE = TRDNBR
```

SMTP alerts

The following stanza must be specified if SMTP messages are to be sent out if the adapter bridge encounters any errors.

Table 27.

Keyword	Mandatory/ Optional/ Default value	Possible values	Description
HostName	Mandatory	The valid URL.	The URL of the SMTP server to be used.
Port	Optional Default: 25	The SMTP port (which usually is 25).	The SMTP port e-mail sender identification.
Sender	Mandatory	Any string.	The e-mail sender identification.
Receivers	Optional Default: FRONT ARENA - WebSphere MQ Bridge Alert!	The valid e-mail receiver addresses.	
Subject	Optional Default: FRONT ARENA - WebSphere MQ Bridge Alert!	Any string.	The e-mail subject.
UserID	Optional		The user ID for mail server.
Password	Optional		The password for the mail server.

Communicating with the WBI adapter

To establish communication between the bridge component and the connector, some configuration properties must fit together. In addition to the communication parameters, the business objects supported by the adapter must correspond to those supported by the bridge.

Event processing

The following event table shows correlations between bridge configuration properties and adapter connector specific properties.

Table 28. Event processing communication

Adapter's connector-specific property	Bridge configuration property (Stanza EventHandler)	Description
HostName	N/A (part of WMQChannel)	The host on which the queue manager resides.
Port	N/A (part of WMQCHannel)	The port the queue manager's listener listens to.
Channel	WMQChannel (Format: <channel>/<transport>/<host name>(<port>)	The channel used for communication with the queue manager.
N/A (part of InputQueue)	WMQQueue Manager	The queue manager used for communication between the bridge and the adapter's connector-specific Property.
InputQueue (Format: queue://<queue manager>/queue)	WMQQueue	The queue used for communication between the bridge and the adapter's connector-specific property.

The example in Table 29 shows consistent adapter application configuration properties and bridge configuration properties.

Table 29. Example properties

Adapter's connector-specific property	Bridge configuration property
HostName=fahost Port=1414	
Channel=CHANNEL1	WMQChannel=CHANNEL1/TCP/fahost(1414) WMQQueueManager=TestQMgr
InputQueue=queue://TestQMgr/FRONTARENA.IN	WMQQueue=FRONTARENA.IN

Request processing

This section shows the properties for request processing.

Table 30. Request processing communication

Adapter's connector-specific property	Bridge configuration property (stanza RequestHandler)	Description
HostName	N/A (part of WMQChannel)	The host on which the queue manager resides.
Port	N/A (part of WMQCHannel)	The port the queue manager's listener listens to.
Channel	Channel WMQChannel (Format: <channel>/<transport>/ <host name>(<port>)	The channel used for communication with the queue manager.
N/A (part of quenames)	WMQQueueManager	The queue manager used for communication between the bridge and the adapter's connector specific-property.
ReplyToQueue (Format: queue://<queue manager>/queue)	WMQReplyQueue	The queue used to send replies (to requests) from the bridge to the adapter's connector specific-property.

Table 31. Attribute and bridge configuration

Parameter of attribute in static conversion meta-object	Bridge Configuration Property (stanza RequestHandler)	Description
OutputQueue (Format: queue://<queue manager>/queue)	WMQRequestQueue	Queue used to send requests from the adapter to the bridge.

Timeouts

When the WBI adapter is set up for synchronous event processing, its static conversion meta-object contains a `ResponseTimeout` value that defines the maximum amount of time the adapter waits for a reply to a service request. The value of bridge configuration parameter `PollFrequency` should be considerably smaller than this value.

WebSphere MQ authorization

To be able to communicate with the WebSphere MQ queue manager, either of the following must apply:

- The user ID under which the bridge is running must be authorized to access the queue manager. This can be accomplished in two ways:
  1. By actually granting access right to this user ID.
  2. By allowing the bridge to put the messages on the queue on behalf of another user ID that is allowed to access the queue manager.

     Note: This is done by setting the property, `MCA User Id`, of the channel used for communication to the second user ID.

- A user ID with the corresponding authorization must be supplied as a value of the either or both of the parameters `WMQUserID` in the EventHandler or the RequestHandler stanza of the bridge configuration file.

Communication with other applications

Any other applications used in conjunction with the adapter must adhere to the message layout rules.

The WebSphere MQ messages created and accepted by the bridge component of the adapter consist of a WebSphere MQ MQMD header followed by the message body. No additional headers or message elements are allowed.

Event notification messages

The following event notification message fields of the MQMD header are set by the bridge component of the adapter for event notification messages.

Table 32. Event notification messages

Field	Description	Values
Format	Message format	There are three values: • `INSERT`, if the event indicates the creation of a new FRONT ARENA object. • `UPDATE`, if the event indicates the modification of an existing FRONT ARENA object. • `DELETE`, if the event indicates the deletion of an existing FRONT ARENA object.
Persistence	Message persistence	`MQPER_AS_Q_DEF`
UserId	User ID of message creator	If a user ID was supplied in the bridge configuration, this value is inserted. Otherwise, the ID that the program is running under is inserted.

Note: As seen in Table 32, the `FORMAT` field is not filled with a predefined WebSphere MQ value, but with the type of operation that caused the originating FRONT ARENA notification to be issued. This piece of information is needed by the Adapter for RONT ARENA which can be used in conjunction with the bridge. As a result, code page and encoding conversion is not supported. The message body contains the FRONT ARENA message read from the AMB - without any modifications.

Request messages

The message body must contain a FRONT ARENA AMB message with all the pieces of information available that the FRONT ARENA application needs to perform the request.

The following header fields of the MQMD header must be set for request messages sent to the bridge component of the adapter.

Table 33. Header field description and value

Field	Description	Value
MessageId	Message identifier	A unique identifier generated by WebSphere MQ when the message is created.

Reply messages

The following fields of the MQMD header are set by the bridge component of the adapter in a reply message.

Table 34. Header field descriptions and values

Field	Description	Values
CorrelationId	Message correlation ID	Filled with the message ID of the corresponding request message.
MessageType	Message type	MQMT_DIAGRAM
Feedback	Feedback code	See the table below, Table 35.

All other header field values are copied from the corresponding request message, except for the message ID, which is newly generated.

Table 35. Adapter feedback codes and descriptions

Adapter feedback code	Description
MQFB_PAN	The `INSERT` or `SELECT` request processing was successful.
MQFB_APPL_FIRST + 2	The `UPDATE` or `DELETE` processing was successful.
MQFB_APPL_FIRST + 4	`SELECT` request: No hit.
MQFB_APPL_FIRST + 6	`SELECT` request: Multiple hits.
MQFB_NAN	Request processing failed.

Note: The feedback codes seen in Table 35are expected by the adapter.

The the message body contains the following parts:

- The body of the request message for `INSERT`, `UPDATE`, or `DELETE` requests which can be processed successfully.
- The FRONT ARENA message received from the AMB for `SELECT` requests, which delivered exactly one hit.
- An error message in any other case.

Chapter 7. Configuring the application

This chapter describes how to configure the FRONT ARENA application. In particular, two components of the application are configured, the FRONT ARENA Message Broker (AMB) and the FRONT ARENA AMB Adapter (AMBA).

Configuring AMB

The stanzas EventHandler and RequestHandler of the bridge component configuration file contain the mandatory property, `Name`. The value of this parameter is used as the client name when connecting to the FRONT ARENA AMB as a reader and writer.

This name must be known to FRONT ARENA in advance. Table system of the AMB database must contain an entry with this name. This entry can be created by issuing an SQL command like the following after connecting to the AMB database. INSERT system (name) VALUES (<client name from the config.file).

Configuring AMBA

The FRONT ARENA AMB Adapter (AMBA) is the FRONT ARENA subsystem that both creates and publishes event notifications and handles any external request messages posted on the AMB. It is highly configurable and must be set up in accordance with the bridge component of the adapter to make both event and request processing work.

Note: For details on how to set the AMBA parameters, see the documentation for the FRONT ARENA application.

Event processing

For event processing, the following AMBA parameters, seen in Table 36, must be set up in a predefined way or their values must be compatible with the configuration parameters of the bridge.

Table 36. AMBA parameters for event processing

AMBA parameter	Value	Description
-sender_source	The value of the AMBA parameter must be the prefix of all topics specified in the bridge parameter value, EventHandler/Filter<n>/Topics and is a valid value for the bridge parameter EventHandler/Filter<n>/Sources	The sender source for outgoing messages. The prefix for all message topics looks as follows: <sender source>/....
-sender_off	0	Disables or enables outgoing messages from AMBA.
-show_changes	0	Include difference information in generated messages.

Request processing

For request processing, the following AMBA parameters, seen in Table 37, must be set up in a predefined way or their values must be compatible with the configuration parameters of the bridge.

Table 37. AMBA parameters for request processing

AMBA parameter	Value	Description
-receiver_sources	The value of the bridge parameter, RequestHandler/ FAPublicationSubject, must be one of the values of this AMBA parameter.	Prefix of topics to listen to; messages are received if their topic looks as follows: <receiver source>/.... (<receiver source> being one of possibly multiple values of this parameter).
-receiver_off	0	Disable or enable the receiving of messages.
-db_writes_off	0	Disable or enable the writing of incoming messages to the FRONT ARENA database.
-force_update	1(*)	Update database records, even if all the values are unchanged.
-show_changes	0(*)	Include difference information in the generated messages.
-sender_source	The value of the bridge parameter, RequestHandler/ FAAMBASenderSource, must be identical with the value of this AMBA parameter (*).	Sender source for outgoing messages; prefix for all message topics which look as follows: <sender source>/....
-sender_off	0(*)	Disable or enable outgoing messages from AMBA.

(*) Parameter values marked in this way are only relevant if synchronous request processing is to be supported.

Message format

The format of the messages accepted and published by the AMBA is highly configurable. The AMBA message configuration must match the message format created and understood by the external application. For example, the configuration must correspond to the business object definitions used by the Adapter for FRONT ARENA in a WebSphere Business Integration scenario.

Chapter 8. Running the adapter

This chapter describes how to install and configure the adapter, and enable the FRONT ARENA application to work with the connector.

Configuring the adapter

Connectors have two types of configuration properties; standard configuration properties and connector-specific configuration properties. Some of these properties have default values that you do not need to change. For others, you must set the values of these properties before you can run the adapter. To configure connector properties with InterChange Server as your integration broker, use Connector Configurator. For more information, refer to Appendix B, "Connector specific properties," on page 125.

A connector obtains its configuration values at startup. During a runtime session, you may want to change the values of one or more connector properties. Changes to some connector configuration properties, such as `AgentTraceLevel`, take effect immediately. Changes to other connector properties require component restart or system restart after a change. To determine whether a property is dynamic (taking effect immediately) or static (requiring either connector component restart or system restart), refer to Appendix B, "Connector specific properties," on page 125.

When you configure connector properties for the Adapter for FRONT ARENA, make sure that the following criteria are met:

- The value specified for the connector property, `HostName`, matches that of the host of your WebSphere MQ server.
- The value specified for the connector property, `Port`, matches that of the port for the listener of the queue manager used by the adapter.
- The value specified for the connector property, `Channel`, matches the server connection channel for the queue manager used by the adapter.
- The values specified for queues must match the names you used in creating the queues.
- The queue URI's for connector properties `InputQueue`, `InProgressQueue`, `ArchiveQueue`, `ErrorQueue`, and `UnsubscribeQueue` must be valid and actually exist. For more information, see "Connector-specific properties" on page 74.

Standard connector properties

Standard configuration properties provide information that all connectors use. See Appendix A, "Standard configuration properties for connectors," on page 107, for documentation of these properties.

Because this adapter supports only InterChange Server as the integration broker, the only configuration properties relevant to it are for InterChange Server.

Connector-specific properties

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

The following table lists the connector-specific configuration properties for the adapter. See the sections that follow for explanations of the properties.

Note: These properties include default queue name values. You will need to change these values to match the queue names you are actually using in your set up.

Table 38. Connector-specific configuration properties

Name	Possible values	Default value	Required
ApplicationPassword	*Login password*		No
ApplicationUserName	*Login user ID*		No
ArchiveQueue	*Queue to which copies of successfully processed messages are sent*	queue://<*queue_manager_name*>/WC_MQCONN.ARCHIVE	No
Channel	*MQ server connector channel*		Yes
ConfigurationMetaObject	*Name of configuration meta-object*		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 unprocessed messages*	queue://<*queue_manager_name*>/WC_MQCONN.ERROR	No
FeedbackCodeMappingMO	Feedback code meta-object		No
HostName	*WebSphere MQ server*		Yes
InDoubtEvents	FailOnStartup Reprocess Ignore LogError	Reprocess	No
InputQueue	*Poll queues*	queue://<*queue_manager_name*>/WC_MQCONN.IN	No
InProgressQueue	*In-progress event queue*	queue://<*queue_manager_name*>/WC_MQCONN.IN_PROGRESS	Yes
PollQuantity	*Number of messages to retrieve from each queue specified in the InputQueue property*	1	No
Port	*Port established for the WebSphere MQ listener*		Yes
ReplyToQueue	*Queue to which response messages are delivered when the adapter issues requests*	queue://<*queue_manager_name*>/WC_MQCONN.REPLYTO	No
UnsubscribedQueue	*Queue to which unsubscribed messages are sent*	queue://<*queue_manager_name*>/WC_MQCONN.UNSUBSCRIBE	No
UseDefaults	true or false	false	

ApplicationPassword
The password used with `UserID` to log in to WebSphere MQ.

Default = None.

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

ApplicationUserName
The user ID used with a `Password` to log in to WebSphere MQ.

Default=None.

If the `ApplicationUserName` is left blank or removed, the adapter uses the default user ID provided by WebSphere MQ.

ArchiveQueue
The queue to which copies of successfully processed messages are sent.

Default = `queue://<queue_manager_name>/WC_MQCONN.ARCHIVE`

Channel
The MQ server adapter channel through which the adapter communicates with WebSphere MQ.

Default=none.

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

ConfigurationMetaObject
The name of the static meta-object containing configuration information for the connector.

Default = none.

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

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

DataHandlerConfigMO
The meta-object passed to the data handler to provide configuration information.

Default = `MO_DataHandler_Default`

DataHandlerMimeType
This allows you to request a data handler based on a particular MIME type. The XML data handler is required for use with FRONT ARENA.

Default = `text/xml`

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

Default = `queue://<queue_manager_name>/WC_MQCONN.ERROR`

FeedbackCodeMappingMO
This property allows you to override and reassign the default feedback codes used to synchronously acknowledge receipt of messages to InterChange Server. This property enables you to specify a meta-object in which each attribute name is understood to represent a feedback code. The corresponding value of the feedback code is the return status that is passed to InterChange Server. For a listing of the default feedback codes, see "Synchronous message delivery" on page 12. The adapter accepts the following attribute values representing WebSphere MQ-specific feedback codes:

- MQFB_APPL_FIRST
- MQFB_APPL_FIRST_OFFSET_*N where N is an integer (interpreted as the value of* MQFB_APPL_FIRST + *N*)

The adapter accepts the following InterChange Server-specific status codes as attribute values in the meta-object:

- SUCCESS
- FAIL
- APP_RESPONSE_TIMEOUT
- MULTIPLE_HITS
- UNABLE_TO_LOGIN
- VALCHANGE
- VALDUPES

The following table shows an example meta-object.

Table 39. Example feedback code meta-object attributes

Attribute name	Default value
MQFB_APPL_FIRST	SUCCESS
MQFB_APPL_FIRST + 1	FAIL
MQFB_APPL_FIRST + 2	UNABLE_TO_LOGIN

Default = none.

HostName
The name of the server hosting WebSphere MQ.

Default=none.

InDoubtEvents
This property specifies how to handle in-progress events that are not fully processed due to unexpected adapter 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
The message queues that will be polled by the adapter for new messages. This parameter should be set to the same value as the EventHandler/WMQQueue parameter in the Bridge for FRONT ARENA configuration file.

If the InputQueue property is not supplied, the connector starts up properly but prints a warning message, and performs request processing only. It will not perform any event processing.

Default= queue://<*queue_manager_name*>/WC_MQCONN.IN

InProgressQueue
Message queue where messages are held during processing.

Default= queue://<*queue_manager_name*>/WC_MQCONN.IN_PROGRESS

PollQuantity
The number of messages to retrieve from each queue specified in the InputQueue property during a pollForEvents scan.

Default =1

Port
The port established for the WebSphere MQ listener.

Default=None.

ReplyToQueue
The queue to which response messages are delivered when the adapter issues requests. This parameter should be set to the same value as the RequestHandler/WMQReplyQueue parameter in the Bridge for FRONT ARENA configuration file.

Default = queue://<*queue_manager_name*>/WC_MQCONN.REPLY

UnsubscribedQueue
The queue to which messages that are not subscribed are sent.

Default = queue://<*queue_manager_name*>/WC_MQCONN.UNSUBSCRIBED

Note: *Always check the values WebSphere MQ provides since they may be incorrect or unknown. If so, please implicitly specify values.

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

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 80.

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 80

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 40..

Table 40. 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 40. Guaranteed-event-delivery connector properties for a connector with a JMS event store (continued)

Connector property	Value
SourceQueue	The 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 the documentation for the *Adapter for WebSphere MQ User Guide*.

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 41 summarizes.

Table 41. 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 40 and Table 41.. To set these connector configuration properties, use the Connector Configurator tool. Connector Configurator displays the connector properties in Table 40 on its Standard Properties tab. It displays the connector properties in Table 41 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 specific properties," on page 125.

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 41 on page 79..
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 InterChange Server 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 42.

Table 42. Guaranteed-event-delivery connector properties for a connector with a non-JMS event store

Connector property	Value
`DeliveryTransport`	`JMS`

Table 42. 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 42. 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 specific properties," on page 125.

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.

Queue uniform resource identifiers (URIs)

The URI for a queue begins with the sequence `queue://` followed by:

- The name of the queue manager on which the queue resides
- Another `/`
- The name of the queue
- Optionally, a list of name-value pairs to set the remaining queue properties

For example, the following URI connects to queue `IN` on queue manager `<queue.manager.name>` and causes all messages to be sent as WebSphere MQ messages with priority 5.

```
queue://<queue.manager.name>/WC_MQCONN.IN?targetClient=1&priority=5
```

The following table shows property names for queue URIs.

Table 43. WebSphere MQ-specific connector property names for queue URIs

Property name	Description	Values
`expiry`	The lifetime of the message in milliseconds.	0 = unlimited positive integers = timeout (in ms).
`priority`	The priority of the message.	0-9, where 1 is the highest priority. A value of -1 means that the property should be determined by the configuration of the queue. A value of -2 specifies that the connector can use its own default value.
`persistence`	Indicates whether the message should be `hardened` to the disk.	1 = non-persistent 2 = persistent A value of -1 means that the property should be determined by the configuration of the queue. A value of -2 specifies that the connector can use its own default value.
`CCSID`	The character set of the destination.	Integers: Valid values are listed in the documentation for the *Adapter for WebSphere MQ User Guide*.

Table 43. WebSphere MQ-specific connector property names for queue URIs (continued)

Property name	Description	Values
`targetClient`	Indicates whether the receiving application is JMS compliant or not.	0 = JMS (MQRFH2 header) 1 = MQ (MQMD header only)
`encoding`	Indicates how to represent numeric fields.	Integers: Valid values are listed in the documentation for the *Adapter for WebSphere MQ User Guide*.

The adapter has no control of the character set (CCSID) or encoding attributes of data in MQMessages. Because data conversion is applied as the data is retrieved from or delivered to the message buffer, the adapter relies upon the IBM WebSphere MQ implementation of JMS to convert data (see the IBM WebSphere MQ Java client library documentation). Accordingly, these conversions should be bi-directionally equivalent to those performed by the native WebSphere MQ API using option `MQGMO_CONVERT`.

The adapter has no control over differences or failures in the conversion process. The adapter can retrieve message data of any CCSID or encoding supported by WebSphere MQ without additional modifications. To deliver a message of a specific `CCSID` or encoding, the output queue must be a fully-qualified URI and specify values for `CCSID` and `encoding`. The adapter passes this information to WebSphere MQ, which (via the JMS API) uses the information when encoding data for MQMessage delivery.

Often, lack of support for CCSID and encoding can be resolved by downloading the most recent version of the IBM WebSphere MQ Java client library from IBM's web site. If problems specific to CCSID and encoding persist, contact Technical Support to discuss the possibility of using an alternate Java Virtual Machine to run the adapter.

Configuring Queue URIs

To configure queues for use with the connector:

- Specify all queues as Uniform Resource Identifiers (URIs). The syntax is:
  `queue://<InterChangeServerName.queue.manager>/<actual queue>`
- Specify the host for the queue manager in connector-specific configuration properties (see Table 38).

Note: If your target application expects an MQMD header only and cannot process the extended MQRFH2 headers used by JMS clients, append `?targetClient=1` to the queue URI.

Meta-objects configuration

The connector uses meta-object entries to determine which business object to associate with a message. The type of business object and verb used in processing an event message is based on the `FORMAT` field contained in the WebSphere MQ message header. You construct a meta-object attribute to store the business object name and verb to associate with the WebSphere MQ message header `FORMAT` field text. Meta-object attributes also contain message processing guidelines.

When a message is retrieved from the input queue, the connector looks up the business object name associated with the FORMAT text field. The message, along with the business object name, is then passed to the data handler. If a business object is successfully populated with message content, the connector checks to see if it is subscribed, and then delivers it to the integration broker using the gotApplEvents() method.

The connector can recognize and read two kinds of meta-objects:

- a static connector meta-object
- a dynamic child meta-object

The attribute values of the dynamic child meta-object duplicate and override those of the static meta-object.

When deciding upon which meta-object will work best for your implementation, consider the following:

- **Static meta-object**
  - Useful if all metadata for different messages is fixed and can be specified at configuration time.
  - Limits you to specifying values by business-object type. For example, all Customer-type objects must be sent to the same destination.
- **Dynamic meta-object**
  - Gives business processes access to information in message headers
  - Allows business processes to change processing of messages at run-time, regardless of business type. For example, a dynamic meta-object would allow you to specify a different destination for every Customer-type object sent to the adapter.
  - Requires changes to the structure of supported business objects—such changes may require changes to maps and business processes.
  - Requires changes to custom data handlers.

Meta-object properties

Table 44 on page 85 provides a complete list of properties supported in meta-objects. Refer to these properties when implementing meta-objects. Your meta object should have one or more of the properties shown in Table 44.

Not all properties are available in both static and dynamic meta-objects. Nor are all properties are readable from or writable to the message header. See the appropriate sections on event and request processing to determine how a specific property is interpreted and used by the connector.

Table 44. WebSphere MQ adapter meta-object properties

Property name	Definable in static meta-object	Definable in dynamic meta-object	Description
`CollaborationName`	Yes	Yes	The `CollaborationName` must be specified in the application specific text of the attribute for the business object/verb combination. For example, if a user expects to handle synchronous event delivery for the business object Customer with the Create verb, the static metadata object must contain an attribute named `Customer_Create`. The `Customer_Create` attribute must contain application specific text that includes a name-value pair. For example, `CollaborationName=MyCustomerProcessingCollab`. See the "Overview of creating static meta-objects" on page 87 section for syntax details. Failure to do this will result in run-time errors when the connector attempts to synchronously process a request involving the Customer business object. **Note:** This property is only available for synchronous requests.
`DataHandlerConfigMO`	Yes	Yes	The meta-object passed to data handler to provide configuration information. If specified in the static meta-object, this will override the value specified in the `DataHandlerConfigMO` connector property. You use this meta-object property when different data handlers are required for processing different business object types. Use the dynamic child meta-object for request processing when the data format may be dependent on the actual business data. The specified business object must be supported by the connector agent. See the description in Appendix B, "Connector specific properties," on page 125.
`DataHandlerMimeType`	Yes	Yes	Allows you to request a data handler based on a particular MIME type. If specified in the meta-object, this will override the value specified in the `DataHandlerMimeType` connector property. Use this meta-object property when different data handlers are required for processing different business object types. Use the dynamic child meta-object for request processing when the data format might be dependent on the actual business data. The business object specified in `DataHandlerConfigMO` should have an attribute that corresponds to the value of this property. See the description in Appendix B, "Connector specific properties," on page 125.
`DataHandlerClassName`	Yes	Yes	See the description in Appendix B, "Connector specific properties," on page 125.

Table 44. WebSphere MQ adapter meta-object properties (continued)

Property name	Definable in static meta-object	Definable in dynamic meta-object	Description
InputFormat	Yes	Yes	The format or type of inbound (event) message to associate with the given business object. This value helps identify the message content and is specified by the application that generated the message. When a message is retrieved and is in this format, it is converted to the given business object, if possible. If this format is not specified for a business object, the connector does not handle subscription deliveries for the given business object. Do not set this property using default meta-object conversion properties; its value is used to match incoming messages to business objects. The field that the connector considers as defining the format in the message can be user-defined via the connector-specific property MessageFormatProperty.
OutputFormat	Yes	Yes	The format to be populated in outbound messages. If the OutputFormat is not specified, the input format is used, if available.
InputQueue	Yes	Yes	The input queue that the connector polls to detect new messages. This property is used to match incoming messages to business objects only. Do not set this property using default conversion properties; its value is used to match incoming messages to business objects. **Note:** The InputQueue property in the connector-specific properties defines which queues the adapter polls. This is the only property that the adapter uses to determine which queues to poll. In the static MO, the InputQueue property and the InputFormat property can serve as criteria for the adapter to map a given message to a specific business object. To implement this feature, you would use connector-specific properties to configure multiple input destinations and optionally map different data handlers to each one based on the input formats of incoming messages. For information, see "Overview of mapping data handlers to input queues" on page 89
OutputQueue	Yes	Yes	The queue to which messages derived from the given business object are delivered.
ResponseTimeout	Yes	Yes	Indicates the length of time in milliseconds to wait before timing out when waiting for a response in synchronous request processing. The connector returns SUCCESS immediately without waiting for a response if this is left undefined or with a value less than zero.
TimeoutFatal	Yes	Yes	Used in synchronous request processing to trigger the connector to return an error message if a response is not received. If this property is True, the connector returns APPRESPONSETIMEOUT to the broker when a response is not received within the time specified by ResponseTimeout. If this property is undefined or set to False, then on a response timeout the connector fails the request but does not terminate. Default = False.

Table 44. WebSphere MQ adapter meta-object properties (continued)

Property name	Definable in static meta-object	Definable in dynamic meta-object	Description
DataEncoding			DataEncoding is the encoding to be used to read and write messages. If this property is not specified in the static meta-object, the connector tries to read the messages without using any specific encoding. DataEncoding defined in a dynamic child meta-object overrides the value defined in the static meta-object. The default value is Text. The format for the value of this attribute is messageType[:*enc*]. I.e., Text:ISO8859_1, Text:UnicodeLittle, Text, or Binary. This property is related internally to the InputFormat property: specify one and only one DataEncoding per InputFormat.
Below are fields mapping specifically to the JMS message header. For specific explanations, interpretation of values, and more, see the JMS API specification. JMS providers may interpret some fields differently so also check your JMS provider documentation for any deviations.			
ReplyToQueue		Yes	The queue to which a response message for a request is to be sent.
Type		Yes	The type of message. The type of message is generally user-definable, depending on JMS provider.
MessageID		Yes	The unique ID for the message (JMS provider specific).
CorrelationID	Yes	Yes	This is used in response messages to indicate the ID of the request message that initiated this response.
Delivery Mode	Yes	Yes	Specifies whether the message is persisted or not in the MOM system. Acceptable values incluce the following: 1=non-persistent 2=persistent Other values, depending on the JMS provider, may be available.
Priority		Yes	The numeric priority of a message. Acceptable values include 0 through 9 inclusive (low to high priority).
Destination		Yes	The current or last (if removed) location of the message in the MOM system.
Redelivered		Yes	Indicates that the JMS provider most likely attempted to deliver the message to the client earlier but the receipt was not acknowledged.
Timestamp		Yes	Indicates that the time message was handed off to the JMS provider.
UserID		Yes	The identity of the user sending the message.
AppID		Yes	The identity of the application sending the message.
DeliveryCount		Yes	The number of delivery attempts.
GroupID		Yes	The identity of the message group.
GroupSeq		Yes	The sequence of this message in the message group specified in the GroupID.
JMSProperties		Yes	See "JMS properties" on page 91.

Overview of creating static meta-objects

The WebSphere MQ adapter configuration meta-object consists of a list of conversion properties defined for different business objects. To view an example static meta-object, launch Business Object Designer and open the following

example that is shipped with the adapter:
connectors\WebSpereMQ\samples\LegacyContact\WebSpereMQ_MO_Config.xsd

The connector supports at most one static meta-object at any given time. You implement a static meta-object by specifying its name for connector property ConfigurationMetaObject.

The structure of the static meta-object is such that each attribute represents a single business object and verb combination and all the metadata associated with processing that object. The name of each attribute should be the name of the business object type and verb separated by an underscore, such as Customer_Create. The attribute application-specific information should consist of one or more semicolon-delimited name-value pairs representing the metadata properties you want to specify for this unique object-verb combination.

Table 45. Static meta-object structure

Attribute name	Application-specific text
<business object type>_<verb>	property=value;property=value;...
<business object type>_<verb>	property=value;property=value;...

For example, consider the following meta-object:

Table 46. Example static meta-object structure

Attribute name	Application-specific information
Customer_Create	OutputFormat=CUST;OutputDestination=QueueA
Customer_Update	OutputFormat=CUST;OutputDestination=QueueB
Order_Create	OutputFormat=ORDER;OutputDestination=QueueC

The meta-object in this example informs the connector that when it receives a request business object of type Customer with verb Create, to convert it to a message with format CUST and then to place it in destination QueueA. If the customer object instead had verb Update, the message would be placed in QueueB. If the object type was Order and had verb Create, the connector would convert and deliver it with format ORDER to QueueC. Any other business object passed to the connector would be treated as unsubscribed.

Optionally, you may name one attribute Default and assign to it one or more properties in the ASI. For all attributes contained in the meta-object, the properties of the default attribute are combined with those of the specific object-verb attributes. This is useful when you have one or more properties to apply universally (regardless of object-verb combination). In the following example, the connector would consider object-verb combinations of Customer_Create and Order_Create as having OutputDestination=QueueA in addition to their individual metadata properties:

Table 47. Example static meta-object structure

Attribute name	Application-specific information
Default	OutputDestination=QueueA
Customer_Update	OutputFormat=CUST
Order_Create	OutputFormat=ORDER

Table 44 on page 85 describes the properties that you can specify as application-specific information in the static meta-object.

Note: If a static meta object is not specified, the connector is unable to map a given message format to a specific business object type during polling. When this is the case, the connector passes the message text to the configured data handler without specifying a business object. If the data handler cannot create a business object based on the text alone, the connector reports an error indicating that this message format is unrecognized.

Configuring a static meta-object

A static meta-object contains application-specific information that you specify about business objects and how the connector processes them. A static meta-object provides the connector with all the information it needs to process a business object when the connector is started.

At the time of implementation, if you know which queues different business objects must be sent to, use a static meta-object. When you create and configure the static meta-object, make sure that the connector subscribes to the static meta-object by specifying the name of the static meta-object in the connector-specific property, `DataHandlerConfigMO`. For more information, see "Connector-specific properties" on page 74.

Steps for creating static meta-objects

1. Launch Business Object Designer. For further information, see the *Business Object Development Guide*.
2. Open the example meta-object `connectors\WBIMB\samples\LegacyItem\Sample_WBIMB_MO_Config.xsd`.
3. Edit the attributes and ASI to reflect your requirements, referring to Table 44 on page 85 and then save the meta-object file.
4. Specify the name of this meta-object file as the value of the connector property, ConfigurationMetaObject.

Overview of mapping data handlers to input queues

You can use the InputQueue property in the application-specific information of the static meta-object to associate a data handler with an input queue. This feature is useful when dealing with multiple trading partners who have different formats and conversion requirements.

Steps for mapping data handlers to input queues

To map a data handler to an InputQueue, perform the following steps:

1. Use connector-specific properties (see "InputQueue" on page 77) to configure one or more input queues.
2. Open the static meta-object in Business Object Designer.
3. For each input queue in the static meta-object, specify the following:
   - queue manager
   - input queue name
   - data handler class name
   - mime type in the application-specific information

For example, the following attribute in a static meta-object associates a data handler with an InputQueue named CompReceipts:

```
[Attribute]
Name = Cust_Create
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = InputQueue=//queue.manager/CompReceipts;DataHandlerClassName=
com.crossworlds.DataHandlers.WBIMB.disposition_notification;DataHandlerMimeType=
message/
disposition_notification
IsRequiredServerBound = false
[End]
```

Overview of creating dynamic child meta-objects

If the connector is required to process a business object differently, depending on the scenario, use a dynamic meta-object. This is a child object that you add to the business object. The dynamic meta-object tells the connector, at run-time, how to process a request. Unlike the static meta-object, which provides the connector with all of the information it needs to process a business object, a dynamic meta-object provides only those additional pieces of logic required to handle the processing for a specific scenario.

Dynamic meta-objects allow you to change the metadata used by the connector to process a business object on a per-request basis during request processing, and to retrieve information about an event message during event processing.

The connector recognizes and reads conversion properties from a dynamic meta-object that is added as a child to the top-level business object passed to the connector. The attribute values of the dynamic child meta-object duplicate the conversion properties that you can specify via the static meta-object that is used to configure the connector.

Since dynamic child meta object properties override those found in static meta-objects, if you specify a dynamic child meta-object, you need not include a connector property that specifies the static meta-object. Accordingly, you can use a dynamic child meta-object independently of the static meta-object and vice-versa.

Table 44 on page 85 describes the properties that you can specify as application-specific information in the dynamic meta-object.

The structure of the dynamic meta-object is such that each attribute represents a single metadata property and value: `meta-object property name =meta-object property value`

Note: All standard IBM WebSphere data handlers are designed to ignore this dynamic meta-object attribute by recognizing the `cw_mo_` tag. You must do the same when developing custom data handlers for use with the adapter.

Creating and configuring dynamic meta-objects

To create and configure a dynamic meta-object, perform the following steps:

- Create the dynamic meta-object and add it as a child to the request business object.

- Program your collaboration with additional logic that populates the dynamic meta-object with information such as the target queue, message format, and other tasks, before issuing it to the connector.

The connector will check for the dynamic meta-object and use its information to determine how to process the business object.

Population of the dynamic child meta-object during polling

In order to provide collaborations with more information regarding messages retrieved during polling, the connector populates specific attributes of the dynamic meta-object, if already defined for the business object created.

Table Table 48 shows how a dynamic child meta-object might be structured for polling.

Table 48. Dynamic child meta-object structure for polling

Property name	Example value
InputFormat	CUST_IN
InputQueue	MYInputQueue
OutputFormat	CxIgnore
OutputQueue	CxIgnore
ResponseTimeout	CxIgnore
TimeoutFatal	CxIgnore

As shown in Table 48, you can define additional attributes, `Input_Format` and `InputQueue`, in a dynamic child meta-object. The `Input_Format` is populated with the format of the message retrieved, while the `InputQueue` attribute contains the name of the queue from which a given message has been retrieved. If these properties are not defined in the child meta-object, they will not be populated.

Example scenario:

- The connector retrieves a message with the format `CUST_IN` from the queue `MyInputQueue`.
- The connector converts this message to a Customer business object and checks the application-specific text to determine if a meta-object is defined.
- If so, the connector creates an instance of this meta-object and populates the `InputQueue` and `InputFormat` attributes accordingly, then publishes the business object to available collaborations.

JMS headers and dynamic child meta-object attributes

You can add attributes to a dynamic meta-object to gain more information about, and more control over, the message transport. This section describes these attributes and how they affect event notification and request processing.

JMS properties: Unlike other attributes in the dynamic meta-object, `JMSProperties` must define a single-cardinality child object. Every attribute in this child object must define a single property to be read/written in the variable portion of the JMS message header as follows:

1. The name of the attribute has no semantic value.
2. The type of the attribute should always be `String` regardless of the JMS property type.

3. The application-specific information of the attribute must contain two name-value pairs defining the name and format of the JMS message property to which the attribute maps. The name is user-definable. The value type must be one of the following:
   - Boolean
   - String
   - Int
   - Float
   - Double
   - Long
   - Short
   - Byte

The table below shows application-specific information properties that you must define for attributes in the `JMSProperties` object.

Table 49. Application-specific information for JMS property attributes

Attribute	Possible values	ASI	Comments
Name	Any valid JMS property name (valid = compatible with type defined in ASI)	`name=<`*`JMS property name`*`>;type=<`*`JMS property type`*`>`	Some vendors reserve certain properties to provide extended functionality. In general, users should not define custom properties that begin with `JMS` unless they are seeking access to these vendor-specific features.
Type	`String`	type=<*see comments*>	This is the type of the JMS property. The JMS API provides a number of methods for setting values in the JMS Message: `setIntProperty`, `setLongProperty`, `setStringProperty`, etc. The type of the JMS property specified here dictates which of these methods is used for setting the property value in the message.

In the example below, a JMSProperties child object is defined for the Customer object to allow access to the user-defined fields of the message header:

```
Customer (ASI = cw_mo_conn=MetaData)
  |-- Id
  |-- FirstName
  |-- LastName
  |-- ContactInfo
  |-- MetaData
        |-- OutputFormat = CUST
```

```
|-- OutputDestination = QueueA
|-- JMSProperties
     |-- RoutingCode = 123 (ASI= name=RoutingCode;type=Int)
     |-- Dept = FD (ASI= name=RoutingDept;type=String)
```

To illustrate another example, Figure 6 shows attribute `JMSProperties` in the dynamic meta-object and definitions for four properties in the JMS message header: ID, GID, RESPONSE and RESPONSE_PERSIST. The application-specific information of the attributes defines the name and type of each. For example, attribute ID maps to JMS property `ID` of type `String`).

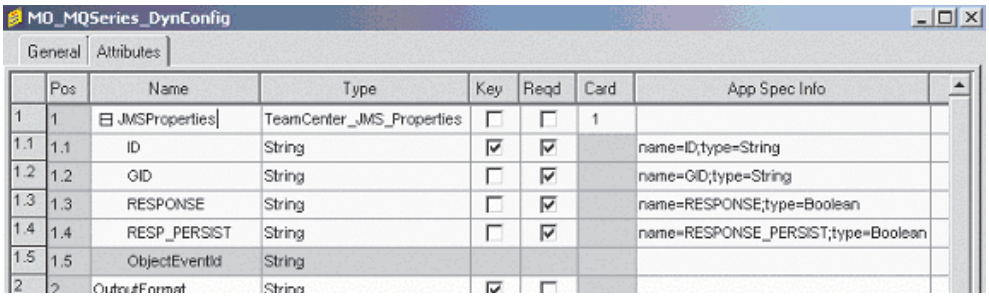

Figure 6. JMS properties attribute in a dynamic meta-object

Steps for creating a dynamic meta-object

To create a dynamic meta-object, perform the following steps:

1. Launch **Business Object Designer**. For further information, see the *Business Object Development Guide*.
2. Open the top-level business object whose processing you want to the dynamic meta-object to influence.
3. Add the dynamic meta-object as a child to your top-level object and include the name-value pair `cw_mo_conn=<MO attribute>` in your top-level object ASI where `<MO attribute>` is the name of the attribute in your top-level object representing the dynamic meta-object. For example:

   ```
   Customer (ASI = cw_mo_conn=MetaData)
     |-- Id
     |-- FirstName
     |-- LastName
     |-- ContactInfo
     |-- MetaData
           |-- OutputFormat = CUST
           |-- OutputDestination = QueueA
   ```

   Upon receipt of a request populated as shown above, the connector would convert the `Customer` object to a message with format `CUST` and then put the message in queue `QueueA`.
4. Save the top-level business object.

Note: Business objects can use the same or different dynamic meta-object or none at all.

Creating multiple instances of the adapter

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."
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*.

Configuring the startup file

Before you start the adapter for FRONT ARENA, you must configure the startup file. Make sure that you modify the start_connector script to point to the location of the client libraries. Do not install multiple versions of the client libraries or versions that are not up to date with your WebSphere MQ server.

Windows

To complete the configuration of the adapter for Windows platforms, you must modify the startup file (either `start_WebSphereMQ.bat` or `start_WebSphereCommerce.bat`,whichever is provided with your adapter):

1. Open the `start_WebSphereCommerceAdapter.bat` file.
2. Scroll to the section beginning with "`Set the directory containing your WebSphere MQ Java client libraries`," and specify the location of your WebSphere MQ Java client libraries.
3. 2. Scroll to the parameter JCLASSES and add the following to the end of the line:

```
;"%CROSSWORLDS%"\DataHandlers\bia_FrontArenaDataHandler.jar
```

UNIX

To complete the configuration of the adapter for UNIX platforms, you must modify the startup file (either `start_WebSphereCommerceAdapter.sh` or `start_WebSphereCommerce.sh`,whichever is provided with your adapter):

1. Open the `start_WebSphereCommerceAdapter.sh` file.
2. Scroll to the section beginning with "`Set the directory containing your WebSphere MQ Java client libraries`," and specify the location of your WebSphere MQ Java client libraries.
3. Scroll to the parameter CLASSPATH and add the following to the end of the line:

```
:${CROSSWORLDS}/DataHandlers/bia_FrontArenaDataHandler.jar
```

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 50 shows.

Table 50. 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 [-cconfigFile ]`
  - 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.

Starting the bridge

The bridge for the FRONT ARENA executable, `bia_bfa`, takes one command line parameter when it is started; the (fully qualified) name of the configuration file to be used.

The application is started by executing the following:

- On Windows: Enter `bia_bfa config.file`.
- On Solaris:
  1. Edit `start_bfa.sh` to specify the name of the configuration file on the `exec` line.
  2. Edit the classpath to add `bia_BFAAlert.jar` in the `CLASSPATH` line.
  3. Run `start_bfa.sh`.

If the configuration file does not exist or cannot be accessed, the bridge immediately terminates, after having displayed an error message. The same is true if any of the mandatory properties are missing or invalid in the configuration file.

When the bridge starts up successfully, proof positive will be indicated by the command line message `Application started up successfully`.

Any problems encountered by the bridge after startup are recorded in the log file and in the trace file (if tracing is switched on). If configured correspondingly, alerting by way of SMTP is supported.

Stopping the bridge

The bridge is stopped by issuing the `stop` command on the bridge command line. Depending on the configuration property, `PollFrequency`, it may take some time for the application to terminate. The bridge cannot stop until the thread interfacing with the FRONT ARENA AMB wakes up and finishes processing messages.

Chapter 9. Troubleshooting the adapter

This chapter describes how to troubleshoot the connector and other scenarios that may be encountered.

Adapter error handling

The bridge component of the adapter can encounter three types of errors:

- The communication with FRONT ARENA fails.
- The communication with WebSphere MQ fails.
- Internal errors, for example, out of memory conditions.

In any case, the errors are logged. Depending on the configuration of the bridge, the bridge component of the adapter uses the FRONT ARENA application alert mechanism and can additionally send SMTP mail to indicate that there is a problem.

In cases of recoverable communication problems, the bridge tries to recover. If recovering fails or if a non-recoverable error occurred, the bridge component of the adapter terminated.

Due to the asynchronous communication between the bridge component and other components in the adapter, a bridge termination will not be noticed directly, except that asynchronous service requests will time out. This can also happen for other reasons, however.

If the bridge is configured to send out an SMTP mail in case of an error, this mail can be received, and follow-on processing can be initiated, by an E-Mail adapter.

Other adapter error handling

The adapter distinguishes three types of errors:

- The conversion of a FRONT ARENA event notification to a business object fails.
- The communication between the adapter and either the bridge component of the adapter or the server-side connector agent fails.
- A service request cannot be converted to a FRONT ARENA message.

In the first case, where the conversion of a FRONT ARENA event notification to a business object fails, the message being processed is forwarded to the adapter's

ErrorQueue if a valid queue is specified in the adapter's application configuration property. Otherwise, the event message is simply discarded.

Besides a corrupted event message, a missing entry for the <object>_<verb> entry in the static conversion meta-object of the adapter can be responsible for this failure.

If the error was detected by the data handler, details about the failure can be found in the adapter's log file.

In the case where the communication between the adapter and either the bridge component of the adapter or the server-side connector agent fails, the adapter creates a log file entry. The event message being processed remains in the adapter's **InProgressQueue**.

In the third case, where a service request cannot be converted to a FRONT ARENA message, a log file entry is created and the erroneous service request is discarded. A negative response is sent to the originating collaboration.

Events that no collaboration has subscribed to are stored in the **UnsubscribedQueue**, if this queue is defined in the adapter configuration. Otherwise, they are simply discarded.

Application tracing

The FRONT ARENA - WebSphere MQ Bridge records all errors encountered in its log file. Depending on the configured log level, major non-error events are reported as well.

In addition, the bridge has a trace which allows tracking its processing in detail. Different trace levels are supported. Both the log and trace level are initially set in the application's configuration file. Both settings can be dynamically modified using commands.

Checking the ErrorQueue

Event notifications that cannot be processed by the adapter are stored in the adapter's ErrorQueue. The adapter does not provide any additional support for handling such errors.

Loss of connection to the application

Since the first version of the adapter supports event processing only, and the communication within the adapter is asynchronous, the adapter will not notice any connection problems. If the bridge cannot re-establish a connection, it will send out an SMTP mail if it is set up accordingly. This mail can be intercepted by an E-Mail adapter, and any follow-on processing can be initiated.

Managing the archive queue

Successfully processed event notifications are stored in the adapter's ArchiveQueue. No additional archiving support is provided. It is up to the customer to save the archive queue's content in a database or in files, if required.

Managing the unsubscribe queue

Events to which no collaboration has subscribed to are stored in the UnsubscribedQueue if the queue is defined in the adapter configuration. Otherwise, they are simply discarded. If these events are to be re-processed at some later point in time, for example, after a collaboration that has subscribed to these events has become active, they must be moved to the InputQueue.

Error handling

All error messages generated by the adapter are stored in a message file named FrontArena.txt. 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 adapter handles specific errors as described in the following sections.

Application timeout

The error message BON_APPRESPONSETIMEOUT is returned when:

- The adapter cannot establish a connection to WebSphere MQ during message retrieval.
- The connector successfully converts a business object to a message but cannot deliver it the outgoing queue due to connection loss.
- The adapter issues a message but times out waiting for a response for a business object with conversion property TimeoutFatal equal to True.
- The adapter receives a response message with a return code equal to APP_RESPONSE_TIMEOUT or UNABLE_TO_LOGIN.

Unsubscribed business object

If the adapter retrieves a message that is associated with an unsubscribed business object, the adapter delivers a message to the queue specified by the UnsubscribedQueue property.

Note: If the UnsubscribedQueue is not defined, unsubscribed messages will be discarded.

Data handler conversion

If the data handler fails to convert a message to a business object, or if a processing error occurs that is specific to the business object (as opposed to WebSphere MQ), the message is delivered to the queue specified by ErrorQueue. If the ErrorQueue is not defined, messages that cannot be processed due to errors will be discarded.

If the data handler fails to convert a business object to a message, BON_FAIL is returned.

Event message to a business object conversion fails

In this case, the message being processed is forwarded to the adapter's ErrorQueue, if a valid queue is specified in the adapter's application configuration property. Otherwise, the event message is simply discarded.

Apart from a corrupted event message, a missing entry for the <object>_<verb> entry in the static conversion meta-object of the adapter can be responsible for this failure. If the error was detected by the data handler, details about the failure can be found in the adapter's log file.

Recommended action: Any message in the error queue requires analysis because it is usually a symptom of a wrong configuration. For example, when the wrong application sends messages to the input queue, or there are errors in the meta-object definition, or the wrong version of message is compared to adapter.

Communication between the adapter and the InterChange Server fails or the adapter crashes

In this case, the adapter creates a log file entry and shuts down. The event message being processed remains in the adapter's *InProgressQueue*. All messages in the *InProgressQueue* have not been processed by the server and are reprocessed automatically, when the adapter is restarted.

Recommended action: None.

Missing subscription

Events that no collaboration has subscribed to are stored in the `UnsubscribedQueue` if this queue is defined in the adapter configuration. Otherwise, they are simply discarded.

Recommended action: If a "sleeping" collaboration was a potential reason for this situation, move the messages to the back to the adapter's input queue to enforce reprocessing after having started the collaboration. Remove the messages with wrong business object-verb combinations (for example, messages without a destination within the integration scenario) from the UnsubscribedQueue and stop the source of these messages.

Problem with event related queues

If one of the event related queues is not operational, the adapter writes a message into the log file and shuts down.

Recommended action: Any termination of the adapter must be analyzed carefully, taking the adapter's log file into consideration.

Duplicate delivery of events

Double, or multiple, delivery of events is very unlikely though not impossible. There is a small gap between the bridge putting an event message onto the adapter's event queue and removing it from FRONT ARENA's AMB. If the bridge crashes between these two neighboring instructions, the event message is re-delivered as soon as the bridge comes up again. The gap cannot be closed by the bridge because FRONT ARENA does not provide any hooks for an external transaction coordinator.

Recommended action: If the bridge delivers an event twice, the adapter is not able to detect this since it does not maintain a history of processed events. The handling of duplicate events must be done in the collaboration in conjunction with the relationship table. FRONT ARENA assigns a unique key to each object created in the system. If the system, and more accuratly, the AMBA, is configured correspondingly, this key is part of the event message that indicates the creation, modification, or deletion of the object. In conjunction with the create or update

time, which is also part of the event message, this piece of information can be used to verify whether the event message had been received before.

Example: For TRADE objects, the message field, `trdnbr`, contains the unique internal FRONT ARENA key, and `update_time` contains the timestamp indicating when the object was created, updated, or deleted.

Request processing

The following section describes issues with request processing.

Adapter not started

When the adapter is not started, the collaboration is not able to process the service request. The flow ends up as a failed flow and needs to be restarted after having started the adapter

Recommended action: Start the adapter and re-start the failed flow thereafter.

Adapter terminates after a request

The adapter terminates after a request, before indicating a reply for the request. This ends up as a failed flow with the adapter indicating that the status of the request is unknown.

Recommended action: Re-start the failed flow after correcting the problem. This can indicate a duplicate request, which can be handled.

Serialization of application-specific business object

When the serialization of the application-specific business object fails (for example, because there is a data handler problem), the adapter throws an exception and writes an entry into its log file. The flow ends up as a failed flow.

Recommended action: Analyze the messages in the log. Then, correct the adapter bug and restart the adapter. Eventually, restart the failed flow.

Request message cannot be put on the request queue

When the message cannot be put onto the request queue, the adapter terminates. Depending on the adapter's configuration, it can restart automatically. The request flow ends up as a failed flow indicating that there was a queue access problem.

Recommended action: Correct the problem with the WebSphere MQ queue and restart the failed flow.

Request-reply processing

The following section describes issues with request-reply processing.

Timeout

If the maximum wait time is exceeded while waiting for the reply to a request, the request is considered as having failed. The timeout is returned to the collaboration and the flows end up as a failed flow.

Although a timeout was encountered, the request may have been successfully processed by the target application. Restarting the failing flow results in a repetition of the request.

Since the adapter does not maintain a request history it is not capable of detecting duplicate requests. FRONT ARENA, duplicate object requests are not a problem. FRONT ARENA internally converts a second create request for an object into an update, which does not apply any changes to the object. A duplicate update request does not have any effect. A second delete request fails because the object no longer exists.

Recommended action: Check to see whether the environment is still up and running, then, restart the failed flow.

Reply message in the reply queue is related to a request that has timed out

If the reply to a request arrives after the maximum wait time configured for the adapter; it is ignored by the adapter. When a reply message in reply queue is related to a request that already timed out a timeout, a timeout message has been sent to the collaboration before, resulting in a failing flow.

Recommended action: Clean up the reply queue regularly. If failed flows exist, it is possible to use these messages for diagnostic purposes.

Reply indicates a failed request

A reply indicating that a failed request has the return information in the feedback field, as well as the error message as part of the message body.

Recommended action: The collaboration must handle the error situations documented. For more information, see Chapter 6, “Configuring the bridge,” on page 57.

Not possible to get a message from the reply queue

When the message cannot be read from the request queue, the adapter terminates. Depending on the adapter’s configuration, it can restart automatically. The request flow ends up as a failed flow indicating that there was a queue access problem.

Recommended action: Correct the problem with the WebSphere MQ queue and restart the failed flows.

Adapter Tracing

Tracing is an optional debugging feature you can turn on to closely follow adapter behavior. Trace messages, by default, are written to STDOUT. See the connector configuration properties for more on configuring trace messages. For more information on tracing, including how to enable and set it, see the *Connector Development Guide*.

What follows is recommended content for trace messages.

Level 0 This level is used for trace messages that identify the adapter version.

Level 1 Use this level for trace messages that provide key information on

each business object processed or record each time a polling thread detects a new message in an input queue.

Level 2 — Use this level for trace messages that log each time a business object is posted to InterChange Server, either from `gotApplEvent()` or `executeCollaboration()`.

Level 3 — Use this level for trace messages that provide information regarding message-to-business-object and business-object-to-message conversions or provide information about the delivery of the message to the output queue.

Level 4 — Use this level for trace messages that identify when the adapter enters or exits a function.

Level 5 — Use this level for trace messages that indicate adapter initialization, represent statements executed in the application, indicate whenever a message is taken off of or put onto a queue, or record business object dumps.

Bridge Tracing

Tracing is an optional debugging feature that you can turn on to closely follow the adapter's behavior. Trace messages are written to the file specified in the Bridge configuration file parameter, `TraceFile`. The trace level can be set in the configuration file parameter, `TraceFile`, or by entering the trace command in the command window. For more information on configuring trace messages, see the bridge configuration properties, Chapter 6, "Configuring the bridge," on page 57.

What follows is recommended content for trace messages.

Level 0 — Trace switched off.

Level 1 — Use this level for trace error messages.

Level 2 — Not used.

Level 3 — Not used.

Level 4 — Use this level for trace messages that identify when the Bridge enters or exits a function.

Level 5 — Use this level for trace messages that indicate informational messages.

Appendix A. Standard configuration properties for connectors

- New and deleted properties
- Configuring standard connector properties
- Summary of standard properties
- Standard configuration properties

New and deleted properties

These standard properties have been added in this release.

New properties

- XMLNameSpaceFormat

Deleted properties

- RestartCount

Configuring standard connector properties

Adapter connectors have two types of configuration properties:

- Standard configuration properties
- Connector-specific configuration properties

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

Using Connector Configurator

You configure connector properties from Connector Configurator, which you access from System Manager. For more information on using Connector Configurator, refer to the 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.
- **Agent restart (ICS only)**
  The change takes effect only after you stop and restart the application-specific component.
- **Component restart**
  The change takes effect only after the connector is stopped and then restarted in System Manager. You do not need to stop and restart the application-specific component or the integration broker.
- **Server restart**
  The change takes effect only after you stop and restart the application-specific component and the integration broker.

To determine how a specific property is updated, refer to the **Update Method** column in the Connector Configurator window, or see the Update Method column in Table 51 on page 108 below.

Summary of standard properties

Table 51 provides a quick reference to the standard connector configuration properties. Not all the connectors make use of all these properties, and property settings may differ from integration broker to integration broker, as standard property dependencies are based on `RepositoryDirectory`.

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

Note: In the "Notes"column in Table 51, the phrase "Repository directory is REMOTE" indicates that the broker is the InterChange Server. When the broker is WMQI or WAS, the repository directory is set to LOCAL

Table 51. Summary of standard configuration properties

Property name	Possible values	Default value	Update method	Notes
AdminInQueue	Valid JMS queue name	*CONNECTORNAME* /ADMININQUEUE	Component restart	Delivery Transport is JMS
AdminOutQueue	Valid JMS queue name	*CONNECTORNAME*/ADMINOUTQUEUE	Component restart	Delivery Transport is JMS
AgentConnections	1-4	1	Component restart	Delivery Transport is MQ or IDL: Repository directory is <REMOTE> (broker is ICS)
AgentTraceLevel	0-5	0	Dynamic	

Table 51. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
ApplicationName	Application name	Value specified for the connector application name	Component restart	
BrokerType	ICS, WMQI, WAS		Component restart	
CharacterEncoding	`ascii7, ascii8, SJIS, Cp949, GBK, Big5, Cp297, Cp273, Cp280, Cp284, Cp037, Cp437` **Note:** This is a subset of supported values.	`ascii7`	Component restart	
ConcurrentEventTriggeredFlows	`1` to `32,767`	`1`	Component restart	Repository directory is <REMOTE> (broker is ICS)
ContainerManagedEvents	`No value` or `JMS`	`No value`	Component restart	Delivery Transport is JMS
ControllerStoreAndForwardMode	`true` or `false`	`true`	Dynamic	Repository directory is <REMOTE> (broker is ICS)
ControllerTraceLevel	`0-5`	`0`	Dynamic	Repository directory is <REMOTE> (broker is ICS)
DeliveryQueue		*CONNECTORNAME*`/DELIVERYQUEUE`	Component restart	JMS transport only
DeliveryTransport	`MQ`, `IDL`, or `JMS`	`JMS`	Component restart	If Repository directory is local, then value is JMS only
DuplicateEventElimination	`true or false`	`false`	Component restart	JMS transport only: Container Managed Events must be <NONE>
FaultQueue		*CONNECTORNAME*`/FAULTQUEUE`	Component restart	JMS transport only
jms.FactoryClassName	`CxCommon.Messaging.jms .IBMMQSeriesFactory` or `CxCommon.Messaging .jms.SonicMQFactory` or any Java class name	`CxCommon.Messaging. jms.IBMMQSeriesFactory`	Component restart	JMS transport only

Table 51. Summary of standard configuration properties (continued)

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

Table 51. Summary of standard configuration properties (continued)

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

Table 51. Summary of standard configuration properties (continued)

Property name	Possible values	Default value	Update method	Notes
RHF2MessageDomain	`mrm, xml`	`mrm`	Component restart	Only if Delivery Transport is JMS and WireFormat is CwXML.
SourceQueue	Valid WebSphere MQ name	*CONNECTORNAME*/SOURCEQUEUE	Agent restart	Only if Delivery Transport is JMS and Container Managed Events is specified
SynchronousRequestQueue		*CONNECTORNAME*/ SYNCHRONOUSREQUESTQUEUE	Component restart	Delivery Transport is JMS
SynchronousRequestTimeout	0 - any number (millisecs)	`0`	Component restart	Delivery Transport is JMS
SynchronousResponseQueue		*CONNECTORNAME*/ SYNCHRONOUSRESPONSEQUEUE	Component restart	Delivery Transport is JMS
WireFormat	`CwXML, CwBO`	`CwXML`	Agent restart	CwXML if Repository Directory is not <REMOTE>: CwBO if Repository Directory is <REMOTE>
WsifSynchronousRequestTimeout	0 - any number (millisecs)	`0`	Component restart	WAS only
XMLNameSpaceFormat	`short, long`	`short`	Agent restart	WebSphere MQ message brokers and WAS only

Standard configuration properties

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

AdminInQueue

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

The default value is `CONNECTORNAME/ADMININQUEUE`.

AdminOutQueue

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

The default value is `CONNECTORNAME/ADMINOUTQUEUE`.

AgentConnections

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

AgentTraceLevel

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

ApplicationName

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

BrokerType

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

CharacterEncoding

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

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

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

There is no default value.

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

- PollQuantity = 1 to 500
- SourceQueue = /SOURCEQUEUE

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

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

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

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

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

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

ControllerStoreAndForwardMode

Applicable only if RepositoryDirectory is <REMOTE>.

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

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

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

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

The default is `true`.

ControllerTraceLevel

Applicable only if `RepositoryDirectory` is <REMOTE>.

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

DeliveryQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default value is `CONNECTORNAME/DELIVERYQUEUE`.

DeliveryTransport

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

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

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

WebSphere MQ and IDL

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

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

JMS

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

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

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

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

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

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

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

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

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

DuplicateEventElimination

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

This property can also be set to `false`.

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

FaultQueue

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

The default value is `CONNECTORNAME/FAULTQUEUE`.

JvmMaxHeapSize

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

The default value is 128m.

JvmMaxNativeStackSize

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

The default value is 128k.

JvmMinHeapSize

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

The default value is 1m.

jms.FactoryClassName

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

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

jms.MessageBrokerName

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

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

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

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

jms.NumConcurrentRequests

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

The default value is 10.

jms.Password

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

There is no default.

jms.UserName

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

There is no default.

ListenerConcurrency

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

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

Locale

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

A locale name has the following format:

ll_TT.codeset

where:

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

By default, only a subset of supported locales appears in the drop-down list. To add other supported values to the drop-down list, you must manually modify the `\Data\Std\stdConnProps.xml` file in the product directory. For more information, 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 in the product directory. Specify the message filename in an absolute path if the message file is not located in the standard location.

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

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

MonitorQueue

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

The default value is CONNECTORNAME/MONITORQUEUE

OADAutoRestartAgent

Valid only when the RepositoryDirectory is <REMOTE>.

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

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

The default value is false.

OADMaxNumRetry

Valid only when the RepositoryDirectory is <REMOTE>.

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

The default value is 1000.

OADRetryTimeInterval

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

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

The default is `10`.

PollEndTime

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

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

PollFrequency

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

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

Set `PollFrequency` to one of the following values:

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

The default is `10000`.

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

PollQuantity

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

FIX

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

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

PollStartTime

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

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

RequestQueue

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

The default value is `CONNECTOR/REQUESTQUEUE`.

RepositoryDirectory

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

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

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

ResponseQueue

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

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

RestartRetryCount

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

The default is `3`.

RestartRetryInterval

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

The default is 1.

RHF2MessageDomain

WebSphere message brokers and WAS only.

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

A sample header would look like this:

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

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

SourceQueue

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

Designates the JMS source queue for the connector framework in support of guaranteed event delivery for JMS-enabled connectors that use a JMS event store. For further information, see “ContainerManagedEvents” on page 114.

The default value is `CONNECTOR/SOURCEQUEUE`.

SynchronousRequestQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default is `CONNECTORNAME/SYNCHRONOUSREQUESTQUEUE`

SynchronousResponseQueue

Applicable only if `DeliveryTransport` is JMS.

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

The default is CONNECTORNAME/SYNCHRONOUSRESPONSEQUEUE

SynchronousRequestTimeout

Applicable only if DeliveryTransport is JMS.

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

The default value is 0.

WireFormat

Message format on the transport.

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

WsifSynchronousRequestTimeout

WAS integration broker only.

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

The default value is 0.

XMLNameSpaceFormat

WebSphere message brokers and WAS integration broker only.

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

The default value is short.

Appendix B. Connector specific properties

This appendix describes connector specific properties.

FRONT ARENA event notification support

If the FRONT ARENA application is set up accordingly, it publishes a message by way of its AMB middleware each time a new object is created or when an existing object is modified or deleted. The bridge component of the adapter is subscribed to all these messages. Each event message received is forwarded to a WebSphere MQ queue.

In order to enable the adapter to handle these events, it must be configured to use this queue as its `InputQueue` (in its connector specific properties).

Note: To make sure that no events are lost, this queue should be set up as a *persistent* queue.

Depending on the settings of the adapter's standard properties, `PollStartTime`, `PollEndTime`, and `PollFrequency` and the value of the connector-specific property `PollQuantity`, entries in this queue are retrieved by the adapter. Having converted the event messages to business objects, they are forwarded to all collaborations that are capable of processing these types of business objects.

Table 52. Connector configuration: Standard properties

Property name	Description	Valid values
PollFrequency	The amount of time between event polling actions.	• Integer value (in milliseconds). • No, to disable polling. • Key, to poll only when the letter p is entered in the connector's command prompt window.
PollStartTime	The time to start polling the event queues.	HH:MM (HH is 0 - 23, MM is 0 - 59) The parameter is disabled if no valid time is specified.
PollEndTime	The time to stop polling the event queues.	HH:MM (HH is 0 - 23, MM is 0 - 59) The parameter is disabled if no valid time is specified.

Table 53. Connector configuration: Connector-specific properties

Property name	Description	Default
InputQueue	One or more event input queues.	N/A

Table 53. Connector configuration: Connector-specific properties (continued)

Property name	Description	Default
PollQuantity	The number of messages to retrieve from each queue specified in the `InputQueue` property during a `pollForEvents` scan.	1

Components

A WebSphere MQ queue is used for notifying the adapter about any events reported by the bridge component of the adapter.

The attribute, `InputQueue`, in the connector specific properties of the adapter, must refer to this queue.

Triggering

Each entry in the adapter's `InputQueue` is considered to be an event. Depending on the value of the adapter's standard property `PollFrequency`, this queue is either polled regularly (property value HH:MM) or polled when explicitly requested by the user (property value KEY). If the queue is polled regularly, a start and end polling time can be configured by means of the standard properties `PollStartTime` and `PollEndTime`.

Create notification

When an event indicating the creation of a new FRONT ARENA object is detected, the adapter tries to create the corresponding business object. The business object name is composed of the value of attribute `BOPrefix` of the data handler meta-object and the object name extracted by the name handler, as described previously.

If an attribute, <business object name>_<verb>, exists in the static configuration meta-object,

- whose <business object name> part matches the business object name (just determined), and
- has the parameter, `InputFormat`, in its application-specific information with a parameter value matching the content of the `FORMAT` field of the event message (INSERT in this case),

then, the business object is created.

The newly created business object verb is the <verb> part of the meta-object attribute. The business object is sent to all interested collaborations or put in the `UnsubscribedQueue` if there are no active subscribers. Otherwise, the event message is rejected and moved to the `ErrorQueue` without being processed.

Table 54. Static configuration meta-object: **MO_FrontArena_DefaultConfig**

Property name	Application-specific information
FA_trade_Create	InputFormat=INSERT;

Update notification

When an event indicating the update of a new FRONT ARENA object is detected, the adapter tries to create the corresponding business object. The business object

name is composed of the value of attribute BOPrefix of the data handler meta-object and the object name extracted by the name handler, as described previously.

If an attribute <business object name>_<verb> exists in the static configuration meta-object,

- whose <business object name> part matches the business object name just determined, and
- has parameter InputFormat in its application-specific information, with a parameter value matching the content of the FORMAT field of the event message (UPDATE in this case),

then, the business object is created.

Its verb is the <verb> part of the meta-object attribute. The business object is sent to all interested collaborations or put in the UnsubscribedQueue if there are no active subscribers.

Otherwise, the event message is rejected and moved to the ErrorQueue without being processed.

Table 55. Static configuration meta-object: **MO_FrontArena_DefaultConfig**

Property name	Application-specific information
FA_trade_Update	InputFormat=UPDATE;

Delete notification

When an event indicating the deletion of a new FRONT ARENA object is detected, the adapter tries to create the corresponding business object. The business object name is composed of the value of attribute BOPrefix of the data handler meta-object and the object name extracted by the name handler, as described previously.

If an attribute <business object name>_<verb> exists in the static configuration meta-object,

- whose <business object name> part matches the business object name just determined, and
- has parameter InputFormat in its application-specific information, with a parameter value matching the content of the FORMAT field of the event message (DELETE in this case),

then, the business object is created.

Its verb is the <verb> part of the meta-object attribute. The business object is sent to all interested collaborations or put in the UnsubscribedQueue if there are no active subscribers.

Table 56. Static configuration meta-object: **MO_FrontArena_DefaultConfig**

Property name	Application-specific information
FA_trade_Delete	InputFormat=DELETE;

Archiving

If a valid WebSphere MQ queue is specified in the adapter's application configuration property, `ArchiveQueue`, any successfully processed events are archived in the queue. Otherwise, no archiving is performed by the adapter.

Note: It is recommended that you clean up the archive queue regularly.

Recovery

The adapter has the standard recovery behavior of the WebSphere MQ adapter. If an event message cannot be processed due to an invalid or unsupported format, it is stored in the `ErrorQueue`, if this queue is defined in the adapter configuration. Otherwise, it is simply discarded.

Any other type of error encountered by the adapter is recorded in the adapter's log file. Depending on the type and gravity of the error, the adapter either retries to perform the operation or terminates.

Unsubscribed events

Events that no collaboration has subscribed to are stored in the `UnsubscribedQueue`, if this queue is defined in the adapter configuration. Otherwise, they are discarded.

If a subscriber becomes active at some later point in time, such events may be re-processed by moving them to the **InputQueue**.

Service requests

The adapter transforms all service requests that it receives from a collaboration to FRONT ARENA messages. These messages are forwarded to the bridge component of the adapter. The `OutputQueue` defined in the static configuration meta-object is used for this purpose. For more information, see "Custom business object handler" on page 32.

The adapter can be set up to handle service requests either synchronously or asynchronously. This is done by either specifying the parameter `ResponseTimeout` in the static configuration meta-object or omitting it. For more information, see "Custom business object handler" on page 32. If the parameter is set, a `ReplyToQueue` must be specified in the connector-specific properties of the adapter.

If the adapter is set up to handle service requests asynchronously, a request is considered to be successful if it can be forwarded to the `OutputQueue`. `BON_SUCCESS` is returned.

In the synchronous mode, the adapter waits for a reply from the bridge component of the adapter for the period of time specified in `ResponseTimeout`. If a reply is received within the given time interval, it is passed back to the calling collaboration. Otherwise, a failure is assumed, and a negative response (`BON_FAIL` is returned to the collaboration.

The bridge component of the adapter can be set up to handle service requests synchronously or asynchronously. The setup of both components must be consistent to make the processing work as expected.

Table 57. Connector configuration: Connector-specific properties

Property name	Description	Default
ReplyToQueue	The queue in which the adapter expects the replies to service requests sent to the bridge component of the adapter.	N/A

Table 58. Static configuration meta-object: **MO_FrontArena_DefaultConfig**

Property name	Application-specific information
Default	**OutputQueue**=queue://DENALI/FRONTARENA.OUT? targetClient=1;**ResponseTimeout**=60000;

Note: It is essential that the `OutputQueue` name is followed by `?targetClient=1`. This ensures that the messages sent to the bridge component of the adapter have the expected format.

Create

Processing of business objects with the Create verb depends on whether the objects are issued asynchronously or synchronously.

Asynchronous processing: This is the default delivery mode for business objects with Create verbs. A message is created from the business object using the data handler and then written to the `OutputQueue` specified in the conversion meta-object. For more information, see "Custom business object handler" on page 32. If the message is delivered, the adapter returns BON_SUCCESS, else BON_FAIL.

Note: With asynchronous processing, the connector has no way of verifying whether the message is received or if action has been taken.

Synchronous processing: If a `ReplyToQueue` has been defined in the connector properties and a `ResponseTimeout` exists in the conversion properties for the business object, the adapter issues a request in synchronous mode. The adapter then waits for a response to verify that appropriate action was taken by the receiving application.

The request message sent to the bridge component of the adapter consists of an MQMD header followed by the message body containing the FRONT ARENA message created from the business object by the data handler.

The MQMD header is initialized based on the information in the following table:

Table 59. Request Message Descriptor Header (MQMD):

Format	Description	Value
Format	The message format.	The value of the parameter, `InputFormat`, in the conversion meta-object attribute belonging to the given <business object name>_<verb> combination. This value can be changed by explicitly setting the corresponding `OutputFormat` parameter. **Note:** The value of this field is ignored by the bridge component of the adapter.
Message type	The message type.	MQMT_DIAGRAM
Report	Options for the report message requested.	When a response message is expected, this field is populated as follows: • MQRO_PAN to indicate that a positive action report is required if processing is successful • MQRO_NAN to indicate that a negative action report is required if processing fails • MQRO_COPY_MSG_ID_TO _CORREL_ID to indicate that the correlation ID of the report generated should equal the message ID of the request originally issued
ReplyToQueue	The name of the reply queue.	When a response message is expected this field is populated with the value of the connector-specific property ReplyToQueue. The value of this field is ignored by the bridge component of the adapter. **Note**: It uses the fixed *reply to* queue defined in its configuration for passing back the request processing results.
Persistence	Message persistence	MQPER_PERSISTENT
Expiry	Message lifetime	MQEI_UNLIMITED

If the business object can be processed, the bridge component of the adapter creates a report message with the feedback field set to `MQFB_PAN`. Additionally, the application populates the message body with a serialized business object containing the attributes of the object just created.

Note: The list of attributes in this response message can differ from that in the request message, however, depending on the FRONT ARENA application configuration.

If the business object cannot be processed, a report message with the feedback field set to `MQFB_NAN` is returned. The message body includes an error message explaining the reason of the failure.

In either case, the application sets the `CorrelationID` field of the reply message to the `MessageID` of the request message and issues it to the queue specified by the `ReplyToQueue` field.

Upon retrieval of a response message, the adapter matches the `correlationID` of the response to the `messageID` of a request message. The adapter then notifies the

thread that issued the request. Depending on the feedback field of the response, the connector either expects a business object or an error message in the message body.

Update

The processing of business objects with the `Update` verb is analogous to the processing of business objects with the `Create` verb.

If the request can be processed successfully, the feedback field in the reply message header is set to `MQFB_APPL_FIRST + 2` (`BON_VALCHANGE`) in this case. Again, the message body contains details about the updated object.

Delete

The processing of business objects with the `Delete` verb is analogous to the processing of business objects with the `Create` verb.

Retrieve

Business objects with the `Retrieve` verb support synchronous processing only. Basically, the adapter processes business objects with this verb as it does for the synchronous processing defined for the `Create`, `Update` and `Delete` verbs. Different feedback codes are returned, however.

Exists

This verb is not supported.

Retrieve by content

This verb is not supported. From the FRONT ARENA application perspective, there is no difference between `Retrieve` and `RetrieveByContent`.

The following table gives an overview of the feedback codes delivered by the bridge component of the adapter.

Table 60. FRONT ARENA - Bridge component feedback codes

Result of processing	Feedback code	Message body
MQFB_PAN	BON_SUCCESS	Insert, Delete, or Retrieve successful.
MQFB_NAN	BON_FAIL	Insert, Update, Delete or Retrieve failed. For Retrieve:Failures other than **multiple hits** and **BO** do not exist.
MQFB_APPL_FIRST + 2	BON_VALCHANGE	The update successful
MQFB_APPL_FIRST + 4	BON_MULTIPLE_ HITS	The retrieve found several matching objects.
MQFB_APPL_FIRST + 6	BON_BO_DOES_ NOT_EXIST	The retrieve did not find any matching objects.

Appendix C. Creating or modifying business objects

This appendix describes information about business objects and their structure.

Overview

The connector comes with example business objects only. The systems integrator, consultant, or customer must build business objects.

The connector is a metadata-driven connector. In WebSphere business integration system 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, object cardinality, the format of the application-specific information, and the database representation of the business object. Therefore, when you create or modify a business object for WebSphere MQ, your modifications must conform to the rules the connector is designed to follow, or the connector cannot process new or modified business objects correctly.

This chapter describes how the connector processes business objects and describes the assumptions the connector makes. You can use this information as a guide to implementing new business objects.

Adapter business object structure

After installing the adapter, you must create business objects. There are no requirements regarding the structure of the business objects other than those imposed by the configured data handler. The business objects that the connector processes can have any name allowed by InterChange Server.

The adapter retrieves messages from a queue and attempts to populate a business object (defined by the meta-object) 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. In fact, there is no business-object level application information. Rather, 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.

Example business object

This section shows an example business object a connector with a Name-Value data handler.

```
[ReposCopy]
Version = 3.0.0
[End]
[BusinessObjectDefinition]
Name = FA_trade
Version = 3.0.0
AppSpecificInfo = TN=trade

    [Attribute]
    Name = creattime
    Type = Date
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=creat_time
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = creatusrnbr
    Type = Integer
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=creat_usrnbr
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = updattime
    Type = Date
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=updat_time
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = updatusrnbr
    Type = Integer
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=updat_usrnbr
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = protection
    Type = Integer
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=protection
    IsRequiredServerBound = false
    [End]
```

```
[Attribute]
Name = ownerusrnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=owner_usrnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = foureyeon
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=four_eye_on
IsRequiredServerBound = false
[End]

[Attribute]
Name = authorizerusrnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=authorizer_usrnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = trdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = prfnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=prfnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = insaddr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
```

```
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=insaddr
IsRequiredServerBound = false
[End]

[Attribute]
Name = instr_insid
Type = String
Cardinality = 1
MaxLength = 39
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=insaddr.insid
IsRequiredServerBound = false
[End]

[Attribute]
Name = instr_instype
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=insaddr.instype
IsRequiredServerBound = false
[End]

[Attribute]
Name = instr_isin
Type = String
Cardinality = 1
MaxLength = 20
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=insaddr.isin
IsRequiredServerBound = false
[End]

[Attribute]
Name = instr_externid1
Type = String
Cardinality = 1
MaxLength = 29
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=insaddr.extern_id1
IsRequiredServerBound = false
[End]

[Attribute]
Name = instr_externid2
Type = String
Cardinality = 1
MaxLength = 29
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=insaddr.extern_id2
IsRequiredServerBound = false
[End]

[Attribute]
```

```
Name = acquireday
Type = Date
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=acquire_day
IsRequiredServerBound = false
[End]

[Attribute]
Name = acquirerptynbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=acquirer_ptynbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = acquirer_ptyid
Type = String
Cardinality = 1
MaxLength = 19
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=acquirer_ptynbr.ptyid
IsRequiredServerBound = false
[End]

[Attribute]
Name = curr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=curr
IsRequiredServerBound = false
[End]

[Attribute]
Name = curr_insid
Type = String
Cardinality = 1
MaxLength = 39
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=curr.insid
IsRequiredServerBound = false
[End]

[Attribute]
Name = valueday
Type = Date
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
```

```
AppSpecificInfo = CN=value_day
IsRequiredServerBound = false
[End]

[Attribute]
Name = time
Type = Date
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=time
IsRequiredServerBound = false
[End]

[Attribute]
Name = quantity
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=quantity
IsRequiredServerBound = false
[End]

[Attribute]
Name = price
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=price
IsRequiredServerBound = false
[End]

[Attribute]
Name = premium
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=premium
IsRequiredServerBound = false
[End]

[Attribute]
Name = fee
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=fee
IsRequiredServerBound = false
[End]

[Attribute]
Name = bofxrate
Type = Double
```

```
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=bo_fx_rate
IsRequiredServerBound = false
[End]

[Attribute]
Name = botrdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=bo_trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = status
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=status
IsRequiredServerBound = false
[End]

[Attribute]
Name = counterpartyptynbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=counterparty_ptynbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = traderusrnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=trader_usrnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = trader_userid
Type = String
Cardinality = 1
MaxLength = 19
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=trader_usrnbr.userid
IsRequiredServerBound = false
```

```
[End]

[Attribute]
Name = optkey1chlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=optkey1_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = optkey2chlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=optkey2_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = optkey3chlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=optkey3_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = optkey4chlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=optkey4_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = hedgetrdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=hedge_trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = trxtrdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
```

```
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=trx_trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = openingbotrdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=opening_bo_trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = optionalkey
Type = String
Cardinality = 1
MaxLength = 31
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=optional_key
IsRequiredServerBound = false
[End]

[Attribute]
Name = type
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=type
IsRequiredServerBound = false
[End]

[Attribute]
Name = marketptynbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=market_ptynbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = origin
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=origin
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = assinf
Type = String
Cardinality = 1
MaxLength = 35
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=assinf
IsRequiredServerBound = false
[End]

[Attribute]
Name = brokerptynbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=broker_ptynbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = bookvalue
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=book_value
IsRequiredServerBound = false
[End]

[Attribute]
Name = bookday
Type = Date
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=book_day
IsRequiredServerBound = false
[End]

[Attribute]
Name = documenttypechlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=document_type_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = guarantorptynbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
```

```
IsRequired = false
AppSpecificInfo = CN=guarantor_ptynbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = rec1accnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=rec1_accnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = rec2accnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=rec2_accnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = pay1accnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=pay1_accnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = pay2accnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=pay2_accnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = calcagent
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=calcagent
IsRequiredServerBound = false
[End]

[Attribute]
Name = cfnetting
```

```
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=cfnetting
IsRequiredServerBound = false
[End]

[Attribute]
Name = creditnetting
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=credit_netting
IsRequiredServerBound = false
[End]

[Attribute]
Name = specialterms
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=special_terms
IsRequiredServerBound = false
[End]

[Attribute]
Name = paymentcount
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=payment_count
IsRequiredServerBound = false
[End]

[Attribute]
Name = text1
Type = String
Cardinality = 1
MaxLength = 29
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=text1
IsRequiredServerBound = false
[End]

[Attribute]
Name = text2
Type = String
Cardinality = 1
MaxLength = 29
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=text2
```

```
IsRequiredServerBound = false
[End]

[Attribute]
Name = yourref
Type = String
Cardinality = 1
MaxLength = 31
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=your_ref
IsRequiredServerBound = false
[End]

[Attribute]
Name = hedgecfwnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=hedge_cfwnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = contracttrdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=contract_trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = clearingtypechlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=clearing_type_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = clearinglocationchlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=clearing_location_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = executiontime
Type = Date
Cardinality = 1
```

```
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=execution_time
IsRequiredServerBound = false
[End]

[Attribute]
Name = netpremium
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=net_premium
IsRequiredServerBound = false
[End]

[Attribute]
Name = aggregate
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=aggregate
IsRequiredServerBound = false
[End]

[Attribute]
Name = aggregatetrdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=aggregate_trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = aggregatepl
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=aggregate_pl
IsRequiredServerBound = false
[End]

[Attribute]
Name = discount
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=discount
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = salescredit
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=sales_credit
IsRequiredServerBound = false
[End]

[Attribute]
Name = salespersonusrnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=sales_person_usrnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = ordnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ordnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = settleseqnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=settle_seqnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = archivestatus
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=archive_status
IsRequiredServerBound = false
[End]

[Attribute]
Name = originalcurr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
```

```
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=original_curr
IsRequiredServerBound = false
[End]

[Attribute]
Name = tradecurr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=trade_curr
IsRequiredServerBound = false
[End]

[Attribute]
Name = odseqnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=od_seqnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = mirrortrdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=mirror_trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = correctiontrdnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=correction_trdnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = haircut
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=haircut
IsRequiredServerBound = false
[End]

[Attribute]
```

```
Name = rightofsubstitution
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=right_of_substitution
IsRequiredServerBound = false
[End]

[Attribute]
Name = salesmargin
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=sales_margin
IsRequiredServerBound = false
[End]

[Attribute]
Name = iasclasschlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ias_class_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = flataccrued
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=flat_accrued
IsRequiredServerBound = false
[End]

[Attribute]
Name = instrument
Type = FA_instrument
ContainedObjectVersion = 3.0.0
Relationship = Containment
Cardinality = N
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=instrument
IsRequiredServerBound = false
[End]

[Attribute]
Name = ObjectEventId
Type = String
Cardinality = 1
MaxLength = 255
IsKey = false
```

```
    IsForeignKey = false
    IsRequired = false
    IsRequiredServerBound = false
    [End]

    [Verb]
    Name = Create
    AppSpecificInfo = FAVERB=insert
    [End]

    [Verb]
    Name = Delete
    AppSpecificInfo = FAVERB=delete
    [End]

    [Verb]
    Name = Retrieve
    AppSpecificInfo = FAVERB=select
    [End]

    [Verb]
    Name = Update
    AppSpecificInfo = FAVERB=update
    [End]
[End]
[BusinessObjectDefinition]
Name = FA_instrument
Version = 3.0.0
AppSpecificInfo = TN=instrument

    [Attribute]
    Name = creattime
    Type = Date
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=creat_time
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = creatusrnbr
    Type = Integer
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=creat_usrnbr
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = updattime
    Type = Date
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=updat_time
    IsRequiredServerBound = false
    [End]

    [Attribute]
```

```
Name = updatusrnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=updat_usrnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = protection
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=protection
IsRequiredServerBound = false
[End]

[Attribute]
Name = ownerusrnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=owner_usrnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = foureyeon
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=four_eye_on
IsRequiredServerBound = false
[End]

[Attribute]
Name = authorizerusrnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=authorizer_usrnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = insid
Type = String
Cardinality = 1
MaxLength = 39
IsKey = false
IsForeignKey = false
IsRequired = false
```

```
AppSpecificInfo = CN=insid
IsRequiredServerBound = false
[End]

[Attribute]
Name = insaddr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = true
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=insaddr
IsRequiredServerBound = false
[End]

[Attribute]
Name = generic
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=generic
IsRequiredServerBound = false
[End]

[Attribute]
Name = notional
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=notional
IsRequiredServerBound = false
[End]

[Attribute]
Name = instype
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=instype
IsRequiredServerBound = false
[End]

[Attribute]
Name = curr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=curr
IsRequiredServerBound = false
[End]

[Attribute]
Name = quotetype
Type = Integer
```

```
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=quote_type
IsRequiredServerBound = false
[End]

[Attribute]
Name = ratetype
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=rate_type
IsRequiredServerBound = false
[End]

[Attribute]
Name = otc
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=otc
IsRequiredServerBound = false
[End]

[Attribute]
Name = mtmfromfeed
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=mtm_from_feed
IsRequiredServerBound = false
[End]

[Attribute]
Name = spotbankingdaysoffset
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=spot_banking_days_offset
IsRequiredServerBound = false
[End]

[Attribute]
Name = externid1
Type = String
Cardinality = 1
MaxLength = 29
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=extern_id1
IsRequiredServerBound = false
```

```
[End]

[Attribute]
Name = externid2
Type = String
Cardinality = 1
MaxLength = 29
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=extern_id2
IsRequiredServerBound = false
[End]

[Attribute]
Name = suspended
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=suspended
IsRequiredServerBound = false
[End]

[Attribute]
Name = productchlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=product_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = issuer
Type = String
Cardinality = 1
MaxLength = 29
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=issuer
IsRequiredServerBound = false
[End]

[Attribute]
Name = issuerptynbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=issuer_ptynbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = contrsize
Type = Double
Cardinality = 1
MaxLength = 255
```

```
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=contr_size
IsRequiredServerBound = false
[End]

[Attribute]
Name = physcontrsize
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=phys_contr_size
IsRequiredServerBound = false
[End]

[Attribute]
Name = undinsaddr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=und_insaddr
IsRequiredServerBound = false
[End]

[Attribute]
Name = undinstype
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=und_instype
IsRequiredServerBound = false
[End]

[Attribute]
Name = monthchar
Type = String
Cardinality = 1
MaxLength = 1
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=month_char
IsRequiredServerBound = false
[End]

[Attribute]
Name = settlement
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=settlement
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = paytype
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=paytype
IsRequiredServerBound = false
[End]

[Attribute]
Name = expday
Type = Date
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=exp_day
IsRequiredServerBound = false
[End]

[Attribute]
Name = expperiodunit
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=exp_period_unit
IsRequiredServerBound = false
[End]

[Attribute]
Name = expperiodcount
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=exp_period_count
IsRequiredServerBound = false
[End]

[Attribute]
Name = paydayoffset
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=pay_day_offset
IsRequiredServerBound = false
[End]

[Attribute]
Name = payperiodunit
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
```

```
IsRequired = false
AppSpecificInfo = CN=pay_period_unit
IsRequiredServerBound = false
[End]

[Attribute]
Name = payperiodcount
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=pay_period_count
IsRequiredServerBound = false
[End]

[Attribute]
Name = calloption
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=call_option
IsRequiredServerBound = false
[End]

[Attribute]
Name = exercisetype
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=exercise_type
IsRequiredServerBound = false
[End]

[Attribute]
Name = strikeprice
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=strike_price
IsRequiredServerBound = false
[End]

[Attribute]
Name = strikecurr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=strike_curr
IsRequiredServerBound = false
[End]

[Attribute]
Name = striketype
```

```
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=strike_type
IsRequiredServerBound = false
[End]

[Attribute]
Name = exotictype
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=exotic_type
IsRequiredServerBound = false
[End]

[Attribute]
Name = digital
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=digital
IsRequiredServerBound = false
[End]

[Attribute]
Name = barrier
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=barrier
IsRequiredServerBound = false
[End]

[Attribute]
Name = payout
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=payout
IsRequiredServerBound = false
[End]

[Attribute]
Name = callable
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=callable
```

```
IsRequiredServerBound = false
[End]

[Attribute]
Name = putable
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=putable
IsRequiredServerBound = false
[End]

[Attribute]
Name = indexfactor
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=index_factor
IsRequiredServerBound = false
[End]

[Attribute]
Name = indextype
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=index_type
IsRequiredServerBound = false
[End]

[Attribute]
Name = roundclean
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=round_clean
IsRequiredServerBound = false
[End]

[Attribute]
Name = roundpremium
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=round_premium
IsRequiredServerBound = false
[End]

[Attribute]
Name = beta
Type = Double
Cardinality = 1
```

```
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=beta
IsRequiredServerBound = false
[End]

[Attribute]
Name = coupons
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=coupons
IsRequiredServerBound = false
[End]

[Attribute]
Name = couprate
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=coup_rate
IsRequiredServerBound = false
[End]

[Attribute]
Name = facevalue
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=face_value
IsRequiredServerBound = false
[End]

[Attribute]
Name = issueday
Type = Date
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=issue_day
IsRequiredServerBound = false
[End]

[Attribute]
Name = lastcoupday
Type = Date
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=last_coup_day
IsRequiredServerBound = false
[End]
```

```
[Attribute]
Name = daycountmethod
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=daycount_method
IsRequiredServerBound = false
[End]

[Attribute]
Name = accruedarrear
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=accrued_arrear
IsRequiredServerBound = false
[End]

[Attribute]
Name = freetext
Type = String
Cardinality = 1
MaxLength = 19
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=free_text
IsRequiredServerBound = false
[End]

[Attribute]
Name = settlecalnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=settle_calnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = openend
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=open_end
IsRequiredServerBound = false
[End]

[Attribute]
Name = categorychlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
```

```
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=category_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = excoupmethod
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ex_coup_method
IsRequiredServerBound = false
[End]

[Attribute]
Name = excoupperiodunit
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ex_coup_period_unit
IsRequiredServerBound = false
[End]

[Attribute]
Name = excoupperiodcount
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ex_coup_period_count
IsRequiredServerBound = false
[End]

[Attribute]
Name = rate
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=rate
IsRequiredServerBound = false
[End]

[Attribute]
Name = refprice
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ref_price
IsRequiredServerBound = false
[End]

[Attribute]
```

```
Name = refvalue
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ref_value
IsRequiredServerBound = false
[End]

[Attribute]
Name = rating1chlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=rating1_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = rating2chlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=rating2_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = rating3chlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=rating3_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = ytmmethod
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=ytm_method
IsRequiredServerBound = false
[End]

[Attribute]
Name = archivestatus
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
```

```
AppSpecificInfo = CN=archive_status
IsRequiredServerBound = false
[End]

[Attribute]
Name = totalissued
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=total_issued
IsRequiredServerBound = false
[End]

[Attribute]
Name = pricedifflimitabs
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=price_diff_limit_abs
IsRequiredServerBound = false
[End]

[Attribute]
Name = pricedifflimitrel
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=price_diff_limit_rel
IsRequiredServerBound = false
[End]

[Attribute]
Name = shortsell
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=short_sell
IsRequiredServerBound = false
[End]

[Attribute]
Name = payoffsetmethod
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=pay_offset_method
IsRequiredServerBound = false
[End]

[Attribute]
Name = startday
Type = Date
```

```
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=start_day
IsRequiredServerBound = false
[End]

[Attribute]
Name = settlecategorychlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=settle_category_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = conversionratio
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=conversion_ratio
IsRequiredServerBound = false
[End]

[Attribute]
Name = originalcurr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=original_curr
IsRequiredServerBound = false
[End]

[Attribute]
Name = combcategorychlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=comb_category_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = pricefindingchlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=price_finding_chlnbr
IsRequiredServerBound = false
```

```
[End]

[Attribute]
Name = isin
Type = String
Cardinality = 1
MaxLength = 20
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=isin
IsRequiredServerBound = false
[End]

[Attribute]
Name = dividendfactor
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=dividend_factor
IsRequiredServerBound = false
[End]

[Attribute]
Name = fixfxrate
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=fix_fx_rate
IsRequiredServerBound = false
[End]

[Attribute]
Name = rebate
Type = Double
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=rebate
IsRequiredServerBound = false
[End]

[Attribute]
Name = exptime
Type = Date
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=exp_time
IsRequiredServerBound = false
[End]

[Attribute]
Name = fixfx
Type = Integer
Cardinality = 1
MaxLength = 255
```

```
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=fix_fx
IsRequiredServerBound = false
[End]

[Attribute]
Name = roundaccrued
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=round_accrued
IsRequiredServerBound = false
[End]

[Attribute]
Name = bondfuturemarket
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=bond_future_market
IsRequiredServerBound = false
[End]

[Attribute]
Name = incomplete
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=incomplete
IsRequiredServerBound = false
[End]

[Attribute]
Name = senioritychlnbr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=seniority_chlnbr
IsRequiredServerBound = false
[End]

[Attribute]
Name = originalinsaddr
Type = Integer
Cardinality = 1
MaxLength = 255
IsKey = false
IsForeignKey = false
IsRequired = false
AppSpecificInfo = CN=original_insaddr
IsRequiredServerBound = false
[End]
```

```
    [Attribute]
    Name = datefrom
    Type = Date
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=date_from
    IsRequiredServerBound = false
    [End]

    [Attribute]
    Name = dateto
    Type = Date
    Cardinality = 1
    MaxLength = 255
    IsKey = false
    IsForeignKey = false
    IsRequired = false
    AppSpecificInfo = CN=date_to
    IsRequiredServerBound = false
    [End]

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

    [Verb]
    Name = Create
    AppSpecificInfo = FAVERB=insert
    [End]

    [Verb]
    Name = Delete
    AppSpecificInfo = FAVERB=delete
    [End]

    [Verb]
    Name = Retrieve
    AppSpecificInfo = FAVERB=select
    [End]

    [Verb]
    Name = Update
    AppSpecificInfo = FAVERB=update
    [End]
[End]
```

Notices

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

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

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

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

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

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

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

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

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

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

Burlingame, CA 94010
U.S.A

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

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

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

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

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

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

COPYRIGHT LICENSE

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

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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

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

WebSphere Business Integration Adapter Framework V2.4.0

Index

G

H

I

J

L

M

N

P

Q

R

S

T

U

V

W

Printed in USA