

*IBM WebSphere Business Integration Collaborations
for UCCnet Item Synchronization Version 4.3.2*



Solution Development Guide

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices and Trademarks” on page 49.

Seventh Edition (January 2004)

This edition applies to Version 4, Release 3, Modification 2 of the *IBM WebSphere Business Integration Collaborations for UCCnet Item Synchronization (5724-H62)*

IBM welcomes your comments. You can send them to the following address:

IBM Canada Ltd. Laboratory
Information Development
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

Include the title and order number of this book, and the page number or topic related to your comment.

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

© **Copyright International Business Machines Corporation 2002, 2004. All rights reserved.**

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

Contents

Solution development guide	1
Introduction	1
Who should read the Solution development guide	1
How the Solution development guide is organized	1
Processing a business object: example workflows (DTD support).	4
ItemAdd workflow: adding a new item to UCCnet (DTD support).	5
ItemPublicationAdd workflow: making a new item available to trading partners and processing their responses.	6
ItemChange workflow: updating item information in UCCnet (DTD support).	8
ItemPublicationChange workflow: making updated item information available to trading partners and processing their responses	10
ItemDelist workflow: making an item permanently unavailable to trading partners (DTD support)	13
ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (DTD support)	14
Processing a business object: example workflows (XSD support)	15
ItemAdd workflow: adding a new item to UCCnet (XSD support)	15
CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses	16
ItemChange workflow: updating item information in UCCnet (schema support)	18
CatalogueItemNotification_Change and CatalogueItemPublication_Change workflow: making updated item information available to trading partners and processing their responses	20
ItemDelist workflow: making an item permanently unavailable to trading partners (schema support)	22
ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (schema support)	23
Maps and data handlers	25

Processing Message Disposition Notifications (MDNs)	32
Retrieving worklists from UCCnet.	32
Customizing the process	32
Sending information from collaboration objects to UCCnet.	33
Receiving data for a collaboration object from UCCnet.	33
Checking that item data exists for fields required by UCCnet.	34
Using the PROCESSED_GTIN table	34
Using the audit_log table.	36
Using the trading_partner table.	37
Using subdiagrams.	38
AUTHORIZATION_RESPONSES subdiagram	39
CATEGORY_ADD_CHANGE subdiagram	39
CATALOGUE_ITEM_CONFIRMATION subdiagram	40
CIN_RESPONSE subdiagram	40
DEAD_LETTER_PUB_RECEIPT subdiagram	40
INITIAL_ITEM_LOAD_REQUEST subdiagram	40
ITEM_ADD_CHANGE subdiagram	41
NEW_ITEM_PUBLICATION_REQUEST subdiagram	41
PUBLICATION_COMMAND_RESPONSE subdiagram	42
RCIR_RESPONSE subdiagram	42
RCIR_QUERY_RESPONSE subdiagram	43
SIMPLE_RESPONSE subdiagram	43
UNKNOWN_MESSAGES subdiagram	43
UNKNOWN_RESPONSE subdiagram	44
Sending email	44
Alerting email recipients of processing errors	44
Sending email through UCCnet_processWorklist collaboration object subdiagrams	45
Logging	47
Tracing	47

Notices and Trademarks	49
Notices	49
Programming interface information	50
Trademarks and service marks	51

Solution development guide

The Item Synchronization for Suppliers solution workflow is composed of business objects, collaboration objects, connectors, and maps. These basic components work together to enable supply-side trading partners to automatically add items to, update or delist items within, or withdraw items from UCCnet[®] when item updates are made in their Enterprise Resource Planning (ERP) applications. When an update is made in a supplier's ERP system, item data is automatically validated, reformatted, and sent to the UCCnet standard registry. Suppliers can also communicate new or updated item information to subscribing trading partners via UCCnet. Thus, enterprise data is synchronized with item data sent outside the enterprise.

Introduction

Who should read the Solution development guide

The Solution development guide describes the internal processing of the Item Synchronization for Suppliers solution. It is intended for programmers who design and implement workflows using the solution and who might participate in designing customizations to this solution. It assumes that users are experienced programmers and that they understand the following concepts and have experience with the software associated with them:

- Developing collaboration objects, business objects, maps, and other related components.
- Installing, configuring, and operating the Item Synchronization for Suppliers solution.

Programmers must also have experience with the operating systems on which their implementations are installed.

How the Solution development guide is organized

The Solution development guide introduces the mechanics of the Item Synchronization for Suppliers solution by first presenting sample, high-level, step-by-step workflows of how the solution handles the following scenarios:

For DTD support:

ItemAdd

Described in the section "ItemAdd workflow: adding a new item to UCCnet (DTD support)" on page 5.

ItemPublicationAdd

Described in the section "ItemPublicationAdd workflow: making a new item available to trading partners and processing their responses" on page 6. (ItemAdd and ItemPublicationAdd workflows normally occur sequentially.)

ItemChange

Described in the section "ItemChange workflow: updating item information in UCCnet (DTD support)" on page 8.

ItemPublicationChange

Described in the section "ItemPublicationChange workflow: making

updated item information available to trading partners and processing their responses” on page 10. (ItemChange and ItemPublicationChange workflows normally occur sequentially.)

ItemDelist

Described in the section “ItemDelist workflow: making an item permanently unavailable to trading partners (DTD support)” on page 13.

ItemWithdrawal

Described in the section “ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (DTD support)” on page 14

For XSD support:

ItemAdd

Described in the section “ItemAdd workflow: adding a new item to UCCnet (XSD support)” on page 15.

CatalogueItemNotification_Add

Used only if UCCnet *is not* used as the data pool as described in the section “CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses” on page 16. (ItemAdd and CatalogueItemNotification_Add workflows normally occur sequentially.)

CatalogueItemPublication_Add

Used only if UCCnet *is* used as the data pool as described in the section “CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses” on page 16 (ItemAdd and CatalogueItemPublication_Add workflows normally occur sequentially.)

ItemChange

Described in the section “ItemChange workflow: updating item information in UCCnet (schema support)” on page 18.

CatalogueItemNotification_Change

Used only if UCCnet *is not* used as the data pool as described in the section “CatalogueItemNotification_Change and CatalogueItemPublication_Change workflow: making updated item information available to trading partners and processing their responses” on page 20. (ItemAdd and CatalogueItemNotification_Add workflows normally occur sequentially.)

CatalogueItemPublication_Change

Used only if UCCnet *is* used as the data pool as described in the section “CatalogueItemNotification_Change and CatalogueItemPublication_Change workflow: making updated item information available to trading partners and processing their responses” on page 20 (ItemAdd and CatalogueItemPublication_Add workflows normally occur sequentially.)

ItemDelist

Described in the section “ItemDelist workflow: making an item permanently unavailable to trading partners (schema support)” on page 22.

ItemWithdrawal

Described in the section “ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (schema support)” on page 23.

Many steps contain links to detailed conceptual information about the mechanics of the solution associated with those steps.

Other sections describe in detail how solution processing operates, as follows:

- “Checking that item data exists for fields required by UCCnet” on page 34 details how a UCCnet_ItemSync collaboration object ensures that the business object to be passed to UCCnet contains data in all of the fields for which UCCnet requires data.
- “Using the PROCESSED_GTIN table” on page 34 describes how the solution populates and maintains data in the provided PROCESSED_GTIN relational table, which permits a UCCnet_processWorklist collaboration object to process incoming INITIAL_ITEM_LOAD_REQUEST commands without the need to communicate with the back-end ERP system.
- “Using the audit_log table” on page 36 provides information on how the solution populates and maintains data in the provided audit_log relational table, used to track events associated with UCCnet activities to support complete end-to-end accountability.
- “Using the trading_partner table” on page 37 identifies how the solution maintains data in the provided trading_partner relational table, which maintains the complete list of trading partners.
- “Retrieving worklists from UCCnet” on page 32 describes how the solution obtains worklists from UCCnet.
- “Using subdiagrams” on page 38 details the logic behind the subdiagrams contained in a UCCnet_processWorklist collaboration object.
- “Sending email” on page 44 describes how solution collaboration objects alert email recipients of processing errors, and how subdiagrams within a UCCnet_processWorklist collaboration object process email for different processing circumstances.
- “Logging” on page 47 describes the capabilities of various collaboration objects to log errors.
- “Tracing” on page 47 outlines how problems that might occur in the solution workflow can be traced and identified.

Planning the configuration

Before you install and configure the Item Synchronization for Suppliers solution, you must determine how you will connect to UCCnet and what message format and protocols you will use.

Connectors:

The way you connect to UCCnet determines the connector that you use to communicate with it. If you exchange messages with UCCnet using an AS2/EDIINT interface protocol, you can use a TPIConnector, an ISoftConnector, or you can use WebSphere Business Integration Connect in conjunction with a JMSConnector. Use the TPI connector if you communicate with UCCnet through Trading Partner Interchange servers. Use the ISoftConnector if you communicate with UCCnet through an iSoft Peer-to-Peer Agent. Use the JMS connector if you communicate with UCCnet through WebSphere Business Integration Connect. If you exchange messages through the UCCnet Command Line Utility (CLU) or are

testing your installation, you can use either a `JTextTPIConnector`, `JTextISoftConnector`, or `JTextJMSConnector`.

Since the actual connector you use is dependent on your set up, this documentation uses “AS2 channel connector” as a general term for any of the `TPIConnector`, `ISoftConnector`, `JMSConnector`, `JTextTPIConnector`, `JTextISoftConnector`, and `JTextJMSConnector`.

Messages:

Messages are exchanged with UCCnet in Extensible Markup Language (XML) documents. The XML document format and the protocol that you select for communication with UCCnet significantly impact the way that you set up your solution. The following options are available:

DTD message format

The format of the XML documents exchanged with UCCnet is defined by a Document Type Definition (DTD). The DTD mode of operation has one protocol available.

Schema message format

The format of the XML documents exchanged by UCCnet is defined by an XML Schema Definition (XSD). The XSD mode of operation has two command protocols available:

CIN operation

The supplier implements its own subscriber data pool. Catalogue_Item_Notification (CIN) messages are sent from the supplier directly to trading partners subscribed to the product categories.

CIP operation

The supplier uses UCCnet as the subscriber data pool. Catalogue_Item (CI) messages containing additional item information that is not included in the UCCnet registry data are sent from the supplier to UCCnet. Catalogue_Item_Publication messages are then sent to UCCnet to identify the subscribers to whom UCCnet needs to send CIN messages.

Processing a business object: example workflows (DTD support)

The information in the following sections outlines at a high level how the Item Synchronization for Suppliers solution handles the workflows that support the DTD-based implementations.

Refer to the Installation guide for detailed information on creating port connections between collaboration objects and between collaboration objects and connectors.

`UCCnet_ItemSync`, `UCCnet_requestWorklist`, `UCCnet_processWorklist`, and `Notify_by_eMail` collaboration objects log error messages if they encounter error situations during any stage of processing. See the section “Logging” on page 47 for detailed information. Tracing can also be enabled for all collaboration objects to record logical flows and data processed. See the section “Tracing” on page 47 for detailed information.

As information moves between collaboration objects, connectors, UCCnet, and other part of the workflow, its format changes. Sometimes, the information will exist as an XML message, sometimes it will be contained in a business object. The conversion from one format to another is usually carried out by maps, which you configure during installation.

The following workflow descriptions discuss the movement of information in general terms. Details about how the information format changes as the information passes between specific points in the workflow is available in the section “Maps and data handlers” on page 25.

ItemAdd workflow: adding a new item to UCCnet (DTD support)

In the ItemAdd workflow, a new item is added to UCCnet. The source of the flow is the creation of a new item in the source ERP application. This workflow does not result in notifications being sent to subscribed demand-side trading partners. Another workflow, detailed in the section “ItemPublicationAdd workflow: making a new item available to trading partners and processing their responses” on page 6, accomplishes sending these notifications.

Notes:

1. Mapping of some attributes in the ItemAdd messages requires the use of value translation tables. The IBM® WebSphere® InterChange Server implements these tables as cross-reference relationships.
2. If you are using XSD support, refer to the documentation found in “ItemAdd workflow: adding a new item to UCCnet (XSD support)” on page 15

The following description shows how high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemAdd workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector specific to that ERP. The connector then converts the item into an application specific business object, passes it through a map to convert it into an ItemBasic generic business object, and then passes the ItemBasic business object to the UCCnet_ItemSync collaboration object with a Create verb.
2. The UCCnet_ItemSync collaboration object accepts the object and checks the required fields for information as detailed in the section “Checking that item data exists for fields required by UCCnet” on page 34. If all required fields are complete, the collaboration object continues processing it. Otherwise, the collaboration object aborts the processing and sends an email, as detailed in the section “Alerting email recipients of processing errors” on page 44. Assume that all fields are complete.
3. The UCCnet_ItemSync collaboration object adds an entry for the new item to the PROCESSED_GTIN table, setting the value for the **withdrawn** field for this entry to N. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.
4. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to record the ItemAdd transaction. See the section “Using the audit_log table” on page 36 for more information about the audit_log table.
5. The UCCnet_ItemSync collaboration objects sends the business object to UCCnet by way of the AS2 channel connector, the AS2 channel server, and the IBM WebSphere Business Integration Data Handler for XML. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
6. UCCnet generates and returns a Message Disposition Notification (MDN) to indicate that it has successfully received the ItemAdd message. See “Processing Message Disposition Notifications (MDNs)” on page 32 for details.

7. UCCnet creates a worklist containing the notification response for a successful Item Add.
8. The supplier sends a worklist request to UCCnet using the JTextRWLConnector and the UCCnet_requestWorklist collaboration object. See “Retrieving worklists from UCCnet” on page 32 for more information
9. UCCnet replies by sending the worklist containing the notification response back to the AS2 channel server.
10. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
11. The UCCnet_processWorklist collaboration object receives the worklist, identifies it as an ItemAdd notification, and dispatches it to its ITEM_ADD_CHANGE subdiagram.

As a result of the ItemAdd workflow, UCCnet has recorded the new item information. Now, the supplier’s demand-side trading partners must be made aware that the item is available. This ongoing workflow, referred to as the ItemPublicationAdd workflow, is continued in the section “ItemPublicationAdd workflow: making a new item available to trading partners and processing their responses.”

ItemPublicationAdd workflow: making a new item available to trading partners and processing their responses

The information in this section describes how the high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemPublicationAdd workflow. In the ItemPublicationAdd workflow, a new item that was passed to UCCnet through the ItemAdd workflow (detailed in the section “ItemAdd workflow: adding a new item to UCCnet (DTD support)” on page 5) is made available to the supplier’s demand-side trading partners. The responses to the new item are processed as well. As a result, the ItemPublicationAdd workflow is described as two subflows:

1. The subflow that makes the new item available to the supplier’s demand-side trading partners, described in the section “ItemPublicationAdd subflow 1: making a new item available to trading partners.”
2. The subflow that processes the demand-side trading partners’ responses to the new item, described in the section “ItemPublicationAdd subflow 2: processing trading partners’ responses to a new item” on page 7.

ItemPublicationAdd subflow 1: making a new item available to trading partners

The following description shows how the ItemPublicationAdd workflow makes a new item available to a supplier’s demand-side trading partners. The source of the flow is the arrival of a UCCnetGBO_envelope at the ITEM_ADD_CHANGE subdiagram of the UCCnet_processWorklist collaboration object as a result of the ItemAdd workflow.

1. The ITEM_ADD_CHANGE subdiagram configures the business object so that the router map in the AS2 channel connector will select the correct transformation map to convert the business object into an ItemPublicationAdd request. See the section “ITEM_ADD_CHANGE subdiagram” on page 41 for more information on this subdiagram.
2. The subdiagram sends the business object to UCCnet by way of the AS2 channel connector, the AS2 channel server, and the IBM WebSphere Business

Integration Data Handler for XML. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.

3. UCCnet generates and returns a Message Disposition Notification (MDN) to indicate that it has successfully received the ItemPublicationAdd message. See “Processing Message Disposition Notifications (MDNs)” on page 32 for details.
4. UCCnet creates a worklist containing the notification response for a successful ItemPublicationAdd.
5. The supplier sends a worklist request to UCCnet using the JTextRWLConnector and the UCCnet_requestWorklist collaboration object. See “Retrieving worklists from UCCnet” on page 32 for more information
6. UCCnet replies by sending the worklist containing the notification response back to the AS2 channel server.
7. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
8. The UCCnet_processWorklist collaboration object receives the worklist, identifies it as a PUB_RELEASE_NEW_ITEM notification, and dispatches it to its NEW_ITEM_PUBLICATION_REQUEST subdiagram.

As a result of this subflow of the ItemPublicationAdd workflow, a new item is now available to a supplier’s demand-side trading partners. The next subflow of the ItemPublicationAdd workflow alerts them of the item’s existence and processes their responses.

ItemPublicationAdd subflow 2: processing trading partners’ responses to a new item

At this point in processing, the first subflow of the ItemPublicationAdd workflow has completed and a UCCnetGBO_envelope business object has arrived in the NEW_ITEM_PUBLICATION_REQUEST subdiagram of the UCCnet_processWorklist collaboration object.

The following describes how the ItemPublicationAdd workflow alerts demand-side trading partners that a new item is available and processes their responses:

1. The NEW_ITEM_PUBLICATION_REQUEST subdiagram verifies that the GTIN value associated with the item is in the PROCESSED_GTIN table and that the item is not withdrawn. It then checks the trading_partner table to verify that there are entries for the trading partner who supplies the new item, and for the demand-side trading partners who will receive notification. See the sections “Using the PROCESSED_GTIN table” on page 34 and “Using the trading_partner table” on page 37 for more information on these tables
2. The subdiagram then configures the business object so that the router map in the AS2 channel connector will select the correct transformation map to convert the business object into an ItemPublicationAdd request.
3. It then sends the business object to UCCnet by way of the AS2 channel connector, the AS2 channel server, and the IBM WebSphere Business Integration Data Handler for XML, and logs the notification in the audit_log table. See “Sending information from collaboration objects to UCCnet” on page 33 and “Using the audit_log table” on page 36 for more details.
4. UCCnet delivers the ItemPublicationAdd to the demand-side trading partners. The trading partners can respond with any of the following responses:

AUTHORIZE

The product information has been integrated into the demand-side trading partner's legacy environment and the demand-side user is ready to begin trading.

PEND_PUBLICATION

The demand-side trading partner is unsure about the proper action to take on the product. The product is being studied, but no action is possible at this time.

REJECT_PUBLICATION

The demand-side trading partner has no interest in the product.

PRE_AUTHORIZATION

The demand-side trading partner wants to begin the process of integrating the product into its legacy environment. For example, this response can indicate that the supplier needs to contact the demand-side user to begin the process of determining order quantities and pricing.

DE_AUTHORIZATION

The demand-side trading partner has removed the product from the assortment and wants no further updates sent for the product.

Note: For this example, assume that the demand-side trading partner responds with an AUTHORIZE response.

5. UCCnet performs a compliance check on the data.
6. If all the data exists in the appropriate formats, then UCCnet creates a worklist for the supplier containing the notification response from the demand-side trading partner.
7. The supplier sends a worklist request to UCCnet using the JTextRWLConnector and the UCCnet_requestWorklist collaboration object. See "Retrieving worklists from UCCnet" on page 32 for more information
8. UCCnet replies by sending the worklist containing the notification response back to the AS2 channel server.
9. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See "Receiving data for a collaboration object from UCCnet" on page 33 for details.
10. The UCCnet_processWorklist collaboration object receives the worklist, identifies it as an AUTHORIZE response, and dispatches it to its AUTHORIZATION_RESPONSES subdiagram. See the section "AUTHORIZATION_RESPONSES subdiagram" on page 39 for more information on this subdiagram.
11. The AUTHORIZATION_RESPONSES subdiagram uses a Notify_by_eMail collaboration object to send email to a predefined list of recipients. It then logs the event in the audit_log table. See the sections "AUTHORIZATION_RESPONSES subdiagram" on page 39, "Sending email through UCCnet_processWorklist collaboration object subdiagrams" on page 45, and "Using the audit_log table" on page 36 for more information.

ItemChange workflow: updating item information in UCCnet (DTD support)

The ItemChange workflow sends updated information about an existing item to UCCnet. The source of the flow is a change to the data of an existing item in the

ERP source application. Issuing a change does not result in notifications being sent to subscribed demand-side trading partners. Another workflow, detailed in the section “ItemPublicationChange workflow: making updated item information available to trading partners and processing their responses” on page 10, accomplishes sending these notifications.

Notes:

1. Mapping of some attributes in the ItemChange messages requires the use of value translation tables. The IBM WebSphere InterChange Server implements these tables as cross-reference relationships.
2. If you are using schema support, refer to the documentation found in “ItemChange workflow: updating item information in UCCnet (schema support)” on page 18

The following description shows how high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemChange workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector specific to that ERP. The connector then converts the item into an application specific business object, passes it through a map to convert it into an ItemBasic generic business object, and then passes the ItemBasic business object to the UCCnet_ItemSync collaboration object with an Update verb.
2. The UCCnet_ItemSync collaboration object accepts the object and checks the required fields for information as detailed in the section “Checking that item data exists for fields required by UCCnet” on page 34. If all required fields are complete, the collaboration object continues processing. Otherwise, the collaboration object aborts the processing and sends an email, as detailed in the section “Alerting email recipients of processing errors” on page 44. Assume that all fields are complete.
3. The UCCnet_ItemSync collaboration object checks that an entry for the item exists in the PROCESSED_GTIN table.

If the item exists in the table and the value for its **withdrawn** field is N, then the collaboration object continues processing.

If the item exists in the table and the value for its **withdrawn** field is Y, then the collaboration object changes the **withdrawn** field value to N, changes the **delete** field value to U, changes the business object verb to *UNWITHDRAWN*, and continues processing.

If the item does not exist in the table, then the collaboration object changes the business object verb to *Create* and adds the item to the PROCESSED_GTIN table, setting the entry’s **withdrawn** field to N.

Note: Assume that the item already exists in the table and is not withdrawn. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.

4. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to identify the ItemChange transaction processed. See the section “Using the audit_log table” on page 36 for more information about the audit_log table.
5. The UCCnet_ItemSync collaboration object sends the information in the business object to UCCnet by way of the AS2 channel connector, AS2 channel server, and the IBM WebSphere Business Integration Data Handler for XML. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.

6. UCCnet generates and returns a Message Disposition Notification (MDN) to indicate that it has successfully received the ItemChange message. See “Processing Message Disposition Notifications (MDNs)” on page 32 for details.
7. UCCnet creates a worklist containing the notification response for a successful ItemChange.
8. The supplier sends a worklist request to UCCnet using the JTextRWLConnector and the UCCnet_requestWorklist collaboration object. See “Retrieving worklists from UCCnet” on page 32 for more information
9. UCCnet replies by sending the worklist containing the notification response back to the AS2 channel server.
10. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
11. The UCCnet_processWorklist collaboration object receives the worklist, identifies it as an ItemChange notification, and dispatches it to its ITEM_ADD_CHANGE subdiagram.

As a result of the ItemChange workflow, UCCnet has updated the item information. Now, the supplier’s demand-side trading partners must be notified that updated information about the item exists. The workflow that accomplishes this is described in the section “ItemPublicationChange workflow: making updated item information available to trading partners and processing their responses.”

ItemPublicationChange workflow: making updated item information available to trading partners and processing their responses

The information in this section describes how the high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemPublicationChange workflow. In the ItemPublicationChange workflow, updated item information that was passed to UCCnet through the ItemChange workflow (detailed in the section “ItemChange workflow: updating item information in UCCnet (DTD support)” on page 8) is made available to the supplier’s demand-side trading partners. The responses to this item information must then be processed. As a result, the ItemPublicationChange workflow is described as two subflows:

- The subflow that makes updated item information available to the supplier’s demand-side trading partners, described in the section “ItemPublicationChange subflow 1: making updated item information available to trading partners.”
- The subflow that processes the demand-side trading partners’ responses to the updated information, described in the section “ItemPublicationChange subflow 2: processing trading partners’ responses to updated item information” on page 11.

ItemPublicationChange subflow 1: making updated item information available to trading partners

The following description shows how the ItemPublicationChange workflow makes updated information available to a supplier’s demand-side trading partners. The source of the flow is the arrival of a UCCnetGBO_envelope at the ITEM_ADD_CHANGE subdiagram of the UCCnet_processWorklist collaboration object as a result of the ItemChange workflow.

1. The ITEM_ADD_CHANGE subdiagram configures the business object so that the router map in the AS2 channel connector will select the correct transformation map to convert the business object into an ItemPublicationChange request. See the section “ITEM_ADD_CHANGE subdiagram” on page 41 for more information on this subdiagram.
2. The subdiagram sends the business object to UCCnet by way of the AS2 channel connector, the AS2 channel server, and the IBM WebSphere Business Integration Data Handler for XML. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
3. UCCnet receives the message requesting it to publish the item to the listed trading partners.
4. UCCnet generates and returns a Message Disposition Notification (MDN) to indicate that it has successfully received the ItemPublicationChange message. See “Processing Message Disposition Notifications (MDNs)” on page 32 for details.
5. UCCnet creates a worklist containing a PUB_RELEASE_DATA_CHANGE notification for a successful ItemPublicationChange.
6. The supplier sends a worklist request to UCCnet using the JTextRWLConnector and the UCCnet_requestWorklist collaboration object. See “Retrieving worklists from UCCnet” on page 32 for more information
7. UCCnet replies by sending the worklist containing the notification response back to the AS2 channel server.
8. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
9. The UCCnet_processWorklist collaboration object receives the worklist, identifies it as a PUB_RELEASE_DATA_CHANGE notification, and dispatches it to its NEW_ITEM_PUBLICATION_REQUEST subdiagram.

As a result of this subflow of the ItemPublicationAdd workflow, a new item is now available to a supplier’s demand-side trading partners. The next subflow of the ItemPublicationAdd workflow alerts them of the item’s existence and processes their responses.

As a result of this subflow of the ItemPublicationChange workflow, updated item information is now available to a supplier’s demand-side trading partners. The next subflow of the ItemPublicationChange workflow alerts them of the updated information and processes their responses to it.

ItemPublicationChange subflow 2: processing trading partners’ responses to updated item information

At this point in processing, the first subflow of the ItemPublicationChange workflow has completed and a UCCnetGBO_envelope business object has arrived in the NEW_ITEM_PUBLICATION_REQUEST subdiagram of the UCCnet_processWorklist collaboration object.

The following description shows how the ItemPublicationChange workflow alerts demand-side trading partners that new item information is available and processes their responses:

1. The NEW_ITEM_PUBLICATION_REQUEST subdiagram verifies that the GTIN value associated with the item is in the PROCESSED_GTIN table and that the item is not withdrawn. It then checks the trading_partner table to verify that there are entries for the trading partner who supplies the new item,

and for the demand-side trading partners who will receive notification. See the sections “Using the PROCESSED_GTIN table” on page 34 and “Using the trading_partner table” on page 37 for more information on these tables

2. The subdiagram then configures the business object so that the router map in the AS2 channel connector will select the correct transformation map to convert the business object into an ItemPublicationChange request.
3. It then sends the business object to UCCnet by way of the AS2 channel connector, the AS2 channel server, and the IBM WebSphere Business Integration Data Handler for XML, and logs the notification in the audit_log table. See “Sending information from collaboration objects to UCCnet” on page 33 and “Using the audit_log table” on page 36 for more details.
4. UCCnet delivers the ItemPublicationAdd to the demand-side trading partners. The trading partners can respond with any of the following responses:

AUTHORIZE

The product information has been integrated into the demand-side trading partner’s legacy environment and the demand-side user is ready to begin trading.

PEND_PUBLICATION

The demand-side trading partner is unsure about the proper action to take on the product. The product is being studied, but no action is possible at this time.

REJECT_PUBLICATION

The demand-side trading partner has no interest in the product.

PRE_AUTHORIZATION

The demand-side trading partner wants to begin the process of integrating the product into its legacy environment. For example, this response can indicate that the supplier needs to contact the demand-side user to begin the process of determining order quantities and pricing.

DE_AUTHORIZATION

The demand-side trading partner has removed the product from the assortment and wants no further updates sent for the product.

Note: For this example, assume that the demand-side trading partner responds with an AUTHORIZE response.

5. UCCnet performs a compliance check on the data. If all the data exists in the appropriate formats, then UCCnet creates a worklist for the supplier containing the notification response from the demand-side trading partner.
6. The supplier sends a worklist request to UCCnet using the JTextRWLConnector and the UCCnet_requestWorklist collaboration object. See “Retrieving worklists from UCCnet” on page 32 for more information.
7. UCCnet replies by sending the worklist containing the notification response to the AS2 channel server.
8. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
9. The UCCnet_processWorklist collaboration object receives the worklist, identifies it as an AUTHORIZE response, and dispatches it to its

AUTHORIZATION_RESPONSES subdiagram. See the section “AUTHORIZATION_RESPONSES subdiagram” on page 39 for more information on this subdiagram.

10. The AUTHORIZATION_RESPONSES subdiagram uses a Notify_by_eMail collaboration object to send email to a predefined list of recipients. It then logs the event in the audit_log table. See the sections “AUTHORIZATION_RESPONSES subdiagram” on page 39, “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45, and “Using the audit_log table” on page 36 for more information.

ItemDelist workflow: making an item permanently unavailable to trading partners (DTD support)

The ItemDelist workflow requests that UCCnet make an item in the repository permanently unavailable. After an item has been delisted, it cannot be returned to active trading. (To remove an item from active trading only temporarily, issue an ItemWithdrawal, as discussed in the section “ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (DTD support)” on page 14.) The source of the flow is the delist of an existing item in the ERP source application. This workflow does not result in notifications being sent to demand-side trading partners.

Note: If you are using schema support, refer to the documentation found in “ItemDelist workflow: making an item permanently unavailable to trading partners (schema support)” on page 22

The following description shows how high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemDelist workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector specific to that ERP. The connector then converts the item into an application specific business object, passes it through a map to convert it to an ItemBasic generic business object, and then passes the ItemBasic business object to the UCCnet_ItemSync collaboration object with a Delist verb.
2. The UCCnet_ItemSync collaboration object accepts the object and checks the required fields for information as detailed in the section “Checking that item data exists for fields required by UCCnet” on page 34. If all required fields are complete, the collaboration object continues processing it. Otherwise, the collaboration object aborts the processing and sends an email, as detailed in the section “Alerting email recipients of processing errors” on page 44. Assume that all fields are complete.
3. The UCCnet_ItemSync collaboration object removes the item from the PROCESSED_GTIN table. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.
4. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to identify the ItemDelist transaction processed. See the section “Using the audit_log table” on page 36 for more information about the audit_log table.
5. The UCCnet_ItemSync collaboration objects sends the business object to UCCnet by way of the AS2 channel connector, the AS2 channel server, and the IBM WebSphere Business Integration Data Handler for XML. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
6. UCCnet generates and returns a Message Disposition Notification (MDN) to indicate that it has successfully received the ItemDelist message. See “Processing Message Disposition Notifications (MDNs)” on page 32 for details.

As a result of the ItemDelist workflow, the item has been permanently delisted in UCCnet and removed from the PROCESSED_GTIN table.

ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (DTD support)

The ItemWithdrawal workflow requests that UCCnet make an item temporarily unavailable to all or selected trading partners. For example, an item can be temporarily removed if it is out of season or not in production. It can also be made available to only a specific set of demand-side trading partners as a special order item. (To remove an item from active trading permanently, issue an ItemDelist, as discussed in the section “ItemDelist workflow: making an item permanently unavailable to trading partners (DTD support)” on page 13.) The source of the flow is the withdrawal of an existing item in the ERP source application. This workflow does not result in notifications being sent to demand-side trading partners.

Note: If you are using schema support, refer to the documentation found in “ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (schema support)” on page 23

The following description shows how high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemWithdrawal workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector specific to that ERP. The connector then converts the item into an application specific business object, passes it through a map to convert it to an ItemBasic generic business object, and then passes the ItemBasic business object to the UCCnet_ItemSync collaboration object with a Withdraw verb.
2. The UCCnet_ItemSync collaboration object accepts the object and checks the required fields for information as detailed in the section “Checking that item data exists for fields required by UCCnet” on page 34. If all required fields are complete, the collaboration object continues processing it. Otherwise, the collaboration object aborts the processing and sends an email, as detailed in the section “Alerting email recipients of processing errors” on page 44. Assume that all fields are complete.
3. The UCCnet_ItemSync collaboration object locates the item in the PROCESSED_GTIN table and sets the value for the **withdrawn** field to Y. This setting prevents an INITIAL_ITEM_LOAD_REQUEST from causing the publication of the item. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.
4. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to identify the ItemWithdrawal transaction processed. See the section “Using the audit_log table” on page 36 for more information about the audit_log table.
5. The UCCnet_ItemSync collaboration objects sends the business object to UCCnet by way of the AS2 channel connector, the AS2 channel server, and the IBM WebSphere Business Integration Data Handler for XML. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
6. UCCnet generates and returns a Message Disposition Notification (MDN) to indicate that it has successfully received the ItemWithdrawal message. See “Processing Message Disposition Notifications (MDNs)” on page 32 for details.

As a result of the ItemWithdrawal workflow, the item has been temporarily withdrawn from UCCnet and has been indicated as withdrawn in the PROCESSED_GTIN table.

Processing a business object: example workflows (XSD support)

ItemAdd workflow: adding a new item to UCCnet (XSD support)

In the ItemAdd workflow, a new item is added to UCCnet. The source of the flow is the creation of a new item in the source ERP application. This workflow does not result in notifications being sent to subscribed demand-side trading partners. Other workflows, detailed in the section “CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses” on page 16, accomplish sending these notifications.

Notes:

1. Mapping of some attributes in the ItemAdd messages requires the use of value translation tables. The IBM WebSphere InterChange Server implements these tables as cross-reference relationships.
2. If you are using DTD support, refer to the documentation found in “ItemAdd workflow: adding a new item to UCCnet (DTD support)” on page 5.

The following description shows how high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemAdd workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector specific to that ERP. The connector converts the item into an application specific business object, passes it through a map to convert it to an ItemBasic generic business object, and then passes the ItemBasic business object to the UCCnet_ItemSync collaboration object with a Create verb.
2. The UCCnet_ItemSync collaboration object accepts the object and checks the required fields for information as detailed in the section “Checking that item data exists for fields required by UCCnet” on page 34. If all required fields are complete, the collaboration object continues processing it. Otherwise, the collaboration object aborts the processing and sends an email, as detailed in the section “Alerting email recipients of processing errors” on page 44. Assume that all fields are complete.
3. The UCCnet_ItemSync collaboration object adds an entry for the new item to the PROCESSED_GTIN table, setting the value for the **withdrawn** field for this entry to N. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.
4. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to record the ItemAdd transaction. See the section “Using the audit_log table” on page 36 for more information about the audit_log table.
5. The UCCnet_ItemSync collaboration objects sends the business object to the ItemCommandRouter collaboration object.
6. The ItemCommandRouter sends the ItemBasic business object containing the RCIR_ADD command to UCCnet by way of the AS2 channel connector and the AS2 channel server. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
7. UCCnet sends an RCIR_ADD_RESPONSE notification to the AS2 channel server indicating a successful item add.
8. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it

to the UCCnet_processWorklist collaboration object. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.

9. The UCCnet_processWorklist collaboration object receives the business object on its From port, identifies it as an RCIR_RESPONSE notification, and dispatches it to its RCIR_RESPONSE subdiagram.

As a result of the ItemAdd workflow, UCCnet has been updated with the new item information. Now, the supplier’s demand-side trading partners must be made aware that the item is available. The workflow that accomplishes this is described in the section “CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses.”

CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses

The information in this section describes how the high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows. In these workflows, a new item that was passed to UCCnet through the ItemAdd workflow (detailed in the section “ItemAdd workflow: adding a new item to UCCnet (XSD support)” on page 15) is made available to the supplier’s demand-side trading partners. The demand-side trading partners’ responses to the new item are processed as well.

When this workflow begins, the ItemAdd workflow has completed and a UCCnetGBO_envelope business object has arrived in the RCIR_RESPONSE subdiagram of the UCCnet_processWorklist collaboration object.

1. The RCIR_RESPONSE subdiagram receives the business object and logs the notification in the audit_log table. See the section “Using the audit_log table” on page 36 for more information on the audit_log table.
2. The subdiagram creates an empty ItemBasic business object and sends it out the DestinationAppRetrieve port with a retrieve verb.
3. A completed ItemBasic object is returned on the same port with a Create verb. This ItemBasic business object is the same one that initiated the RCIR command, which was sent to UCCnet in the ItemAdd workflow.
- 4.

When using a supplier-implemented data source pool (CIN operation):

- a. The subdiagram sends the ItemBasic business object out the RCIR_RESPONSE port to the CIN_CIP_Dispatcher collaboration object.
- b. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port and maps it to a CatalogueItemNotification_ADD UCCnetGBO_envelope business object.
- c. It uses the category code from the business object to retrieve the GLNs of any trading partners that subscribe to the category. The GLN information is taken from the GLN subscription file determined by the DISPATCHER_GLN_FILE property.
- d. For each GLN found in the subscription file, the collaboration object sends a CatalogueItemNotification_ADD (CIN_ADD) notification to the AS2 channel connector for delivery to UCCnet. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.

- e. UCCnet receives the CIN_ADD notifications as XML messages and forwards them to the demand-side trading partners.

When using UCCnet as the data source pool (CIP operation):

- a. The subdiagram sends the ItemBasic business object out the RCIR_RESPONSE port to a second instance of the AS2 channel connector where it is mapped to a PublicationCommand CatalogueItem_ADD and sent to UCCnet by way of the AS2 channel server. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
- b. UCCnet receives the CI_ADD request and returns a CatalogueItem response notification to the AS2 channel server.
- c. The AS2 channel server delivers the response to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
- d. The UCCnet_processWorklist collaboration object, identifies the returned business object as a CI response notification, and sends it to the PUBLICATION_COMMAND_RESPONSE subdiagram of the UCCnet_processWorklist collaboration object.
- e. The subdiagram creates an empty ItemBasic business object and sends it to the DestinationAppRetrieve port with a retrieve verb. An ItemBasic business object with a Create verb is returned on the same port. This ItemBasic business object is the same one that initiated the RCIR command, which was sent to UCCnet in the ItemAdd workflow.
- f. The subdiagram sends the business object to the CIN_CIP_Dispatcher collaboration object.
- g. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port and maps it to a CatalogueItemPublication_ADD UCCnetGBO_envelope business object.
- h. It uses the category code from the business object to retrieve the GLNs of any trading partners that subscribe to the category from the GLN subscription file, defined by the DISPATCHER_GLN_FILE property.
- i. For each GLN that the CIN_CIP_Dispatcher collaboration object finds in the subscription file, it sends a CatalogueItemPublication_ADD (CIP_ADD) notification to the AS2 channel connector for delivery to UCCnet. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
- j. UCCnet sends out a CatalogueItemNotification to each subscribing demand-side trading partner for whom a CIP was received.
- k. UCCnet returns a CatalogueItemPublication response to the AS2 channel server.
- l. The AS2 channel server delivers the response to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
- m. The UCCnet_processWorklist collaboration object receives the business object, identifies it as a CIP response notification, and sends it to the PUBLICATION_COMMAND_RESPONSE subdiagram of the UCCnet_processWorklist collaboration object.

- n. The PUBLICATION_COMMAND_RESPONSE subdiagram sends the business object to a Notify_by_eMail collaboration object with a Create verb to send a notification email to the supplier. See “PUBLICATION_COMMAND_RESPONSE subdiagram” on page 42 for more information on this subdiagram.
5. At this point, the trading partners can respond to UCCnet with any of the following Catalogue Item Confirmation responses:

REVIEW

The retailer is reviewing the item.

REJECTED

The retailer has rejected the item. No additional information is requested at this time.

ACCEPTED

The retailer has accepted the item, but has not yet synchronized it. This state is similar to the DTD-based PRE-AUTHORIZATION state.

SYNCHRONISED

The retailer has accepted the item and synchronized it. This state is similar to a DTD-based AUTHORIZE state.

The rest of this example assumes that a demand-side trading partner has responded with a SYNCHRONISED response.

6. UCCnet performs a compliance check on the data. If all the data exists in the appropriate format UCCnet creates a worklist for the supplier containing this notification response from the demand-side trading partner.
7. The supplier sends a worklist request to UCCnet using the JTextRWLConnector and the UCCnet_requestWorklist collaboration object. See “Retrieving worklists from UCCnet” on page 32 for more information
8. UCCnet replies by sending the worklist containing the notification response back to the AS2 channel server.
9. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
10. The UCCnet_processWorklist collaboration object receives the response, identifies it as a CATALOGUE_ITEM_CONFIRMATION response, and dispatches it to its CATALOGUE_ITEM_CONFIRMATION subdiagram.
11. The CATALOGUE_ITEM_CONFIRMATION subdiagram uses a Notify_by_eMail collaboration object to send email to a predefined list of recipients. It then logs the event in the audit_log table. See the sections “CATALOGUE_ITEM_CONFIRMATION subdiagram” on page 40, “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45, and “Using the audit_log table” on page 36 for more information.

ItemChange workflow: updating item information in UCCnet (schema support)

The ItemChange workflow sends updated information about an existing item to UCCnet. The source of the flow is a change to the data of an existing item in the ERP source application. Issuing a change does not result in notifications being sent to subscribed demand-side trading partners. Other workflows, detailed in the section “CatalogueItemNotification_Change and CatalogueItemPublication_Change

workflow: making updated item information available to trading partners and processing their responses” on page 20, accomplish sending these notifications.

Notes:

1. Mapping of some attributes in the ItemChange messages requires the use of value translation tables. The IBM WebSphere InterChange Server implements these tables as cross-reference relationships.
2. If you are not using schema support, refer to the documentation found in “ItemChange workflow: updating item information in UCCnet (DTD support)” on page 8.

The following description shows how high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemChange workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector specific to that ERP. The connector converts the item into an application specific business object (ASBO), passes it through a map to convert it to an ItemBasic generic business object (GBO), and then passes this ItemBasic business object to the UCCnet_ItemSynch collaboration object with an Update verb.
2. The UCCnet_ItemSynch collaboration object accepts the object and checks the required fields for information as detailed in the section “Checking that item data exists for fields required by UCCnet” on page 34. If all required fields are complete, the collaboration object continues processing it. Otherwise, the collaboration object aborts the processing and sends an email, as detailed in the section “Alerting email recipients of processing errors” on page 44. Assume that all fields are complete.
3. The UCCnet_ItemSynch collaboration object checks that an entry for the item exists in the PROCESSED_GTIN table.

If the item exists in the table and the value for its **withdrawn** field is N, then the collaboration object continues processing.

If the item exists in the table and the value for its **withdrawn** field is Y, then the collaboration object changes the **withdrawn** field value to N, changes the **delete** field value to U, changes the business object verb to Unwithdrawn, and continues processing.

If the item does not exist in the table, then the collaboration object changes the business object verb to *Create* and adds the item to the PROCESSED_GTIN table, setting the entry’s **withdrawn** field to N.

Note: Assume that the item already exists in the table and is not withdrawn. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.

4. The UCCnet_ItemSynch collaboration object adds an entry to the audit_log table to identify the ItemChange transaction processed. See the section “Using the audit_log table” on page 36 for more information about the audit_log table.
5. The UCCnet_ItemSynch collaboration object sends the business to the ItemCommandRouter collaboration object.
6. The ItemCommandRouter collaboration sends the ItemBasic business object containing the RCIR_CHANGE command to UCCnet by way of the AS2 channel connector and the AS2 channel server. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
7. UCCnet sends an RCIR_CHANGE_RESPONSE notification to the AS2 channel server indicating a successful item change.

8. The AS2 channel server delivers the response to the AS2 channel connector, which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
9. The UCCnet_processWorklist collaboration object receives the business object, identifies it as a RCIR_CHANGE_RESPONSE notification, and dispatches it to its RCIR_RESPONSE subdiagram.

As a result of the ItemChange workflow, UCCnet has been updated with the new item information. Now, the supplier’s demand-side trading partners must be notified that the item information is available. The workflows that accomplish this are detailed in the section “CatalogueItemNotification_Change and CatalogueItemPublication_Change workflow: making updated item information available to trading partners and processing their responses.”

CatalogueItemNotification_Change and CatalogueItemPublication_Change workflow: making updated item information available to trading partners and processing their responses

The information in this section describes how the high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the CatalogueItemNotification_Change and CatalogueItemPublication_Change workflows. In these workflows, updated item information that was passed to UCCnet through the ItemChange workflow (detailed in the section “ItemChange workflow: updating item information in UCCnet (schema support)” on page 18) is made available to the supplier’s demand-side trading partners. The demand-side trading partners’ responses to this item information must then be processed.

When this workflow begins, the ItemChange workflow has completed and a UCCnetGBO_envelope business object has arrived in the RCIR_RESPONSE subdiagram of the UCCnet_processWorklist collaboration object.

1. The RCIR_RESPONSE subdiagram receives the business object and logs the notification in the audit_log table. See the section “Using the audit_log table” on page 36 for more information on the audit_log table.
2. The subdiagram creates an empty ItemBasic business object and sends it out the DestinationAppRetrieve port with a retrieve verb.
3. A completed ItemBasic object is returned on the same port with an Update verb. This ItemBasic business object is the same as the one that initiated the corresponding RCIR command which was sent to UCCnet.
- 4.

When using a supplier-implemented data source pool (CIN operation):

- a. The subdiagram sends the ItemBasic business object out the RCIR_RESPONSE port to the CIN_CIP_Dispatcher collaboration object.
- b. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port and maps it to a CatalogueItemNotification_CHANGE UCCnetGBO_envelope business object.
- c. It uses the category code from the business object to retrieve the GLNs of any trading partners that subscribe to the category. The GLN information is taken from the GLN subscription file determined by the DISPATCHER_GLN_FILE property.

- d. For each GLN found in the subscription file, the collaboration object sends a CatalogueItemNotification_CHANGE (CIN_CHANGE) notification to the AS2 channel connector for delivery to UCCnet. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
- e. UCCnet receives the CIN_CHANGE notifications as XML messages and forwards them to the demand-side trading partners.

When using UCCnet as the data source pool (CIP operation):

- a. The subdiagram sends the ItemBasic business object out the RCIR_RESPONSE port to a second instance of the AS2 channel connector where it is mapped to a PublicationCommand CatalogueItem_CHANGE and sent to UCCnet by way of the AS2 channel server. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
 - b. UCCnet receives the CI_Change request and returns a CatalogueItem response notification to the AS2 channel server.
 - c. The AS2 channel server delivers the worklist to the AS2 channel connector, which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
 - d. The UCCnet_processWorklist collaboration object, identifies the returned business object as a CI response notification, and sends it to the PUBLICATION_COMMAND_RESPONSE subdiagram of the UCCnet_processWorklist collaboration object.
 - e. The subdiagram creates an empty ItemBasic business object and sends it to the DestinationAppRetrieve port with a Retrieve verb. An ItemBasic business object with an Update verb is returned on the same port. This ItemBasic business object is the same one that initiated the RCIR command, which was sent to UCCnet in the ItemChange workflow.
 - f. The subdiagram sends the business object to a Notify_by_eMail collaboration object in order to initiate the sending of a notification email to the suppliers.
 - g. The list of demand-side trading partners to receive a notification is determined by UCCnet based on the CatalogueItemPublication (CIP) requests sent to UCCnet when the catalogue item was originally added. UCCnet sends out a CatalogueItemNotification to each subscribing demand-side trading partner for whom a CIP was previously received.
5. At this point, the trading partners can respond with any of the following Catalogue Item Confirmation responses:

REVIEW

The retailer is reviewing the item.

REJECTED

The retailer has rejected the item. No additional information is requested at this time.

ACCEPTED

The retailer has accepted the item, but has not yet synchronized it. This state is similar to the DTD-based PRE-AUTHORIZATION state.

SYNCHRONISED

The retailer has accepted the item and synchronized it. This state is similar to a DTD-based AUTHORIZE state.

The rest of this example assumes that a demand-side trading partner has responded with a SYNCHRONISED response.

6. UCCnet performs a compliance check on the data. If all the data exists in the appropriate format, then UCCnet creates a worklist for the supplier containing the SYNCHRONISED notification response from the demand-side trading partner.
7. The supplier sends a worklist request to UCCnet using the JTextRWLConnector and the UCCnet_requestWorklist collaboration object. See “Retrieving worklists from UCCnet” on page 32 for more information
8. UCCnet replies by sending the worklist containing the notification response back to the AS2 channel server.
9. The AS2 channel server delivers the worklist to the AS2 channel connector which transforms it into a UCCnetGBO_envelope business object and delivers it to the UCCnet_processWorklist collaboration object. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
10. The UCCnet_processWorklist collaboration object receives the worklist, identifies it as a CATALOGUE_ITEM_CONFIRMATION response, and dispatches it to its CATALOGUE_ITEM_CONFIRMATION subdiagram.
11. The CATALOGUE_ITEM_CONFIRMATION subdiagram uses a Notify_by_eMail collaboration object to send email to a predefined list of recipients. It then logs the event in the audit_log table. See the sections “CATALOGUE_ITEM_CONFIRMATION subdiagram” on page 40, “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45, and “Using the audit_log table” on page 36 for more information.

ItemDelist workflow: making an item permanently unavailable to trading partners (schema support)

The ItemDelist workflow requests that UCCnet make an item in the repository permanently unavailable. After an item has been delisted, it cannot be returned to active trading. (To remove an item from active trading only temporarily, issue an ItemWithdrawal, as discussed in the section “ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (schema support)” on page 23.) The source of the flow is the delist of an existing item in the ERP source application. This workflow does not result in notifications being sent to demand-side trading partners.

Note: If you are not using schema support, refer to the documentation found in “ItemDelist workflow: making an item permanently unavailable to trading partners (DTD support)” on page 13

The following description shows how high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemDelist workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector specific to that ERP. The connector converts the item into an application specific business object, and passes it through a map to convert it to an ItemBasic generic business object, and then passes the ItemBasic business object to the UCCnet_ItemSync collaboration object with a Delist verb.
2. The UCCnet_ItemSync collaboration object accepts the object and checks the required fields for information as detailed in the section “Checking that item data exists for fields required by UCCnet” on page 34. If all required fields are complete, the collaboration object continues processing it. Otherwise, the

collaboration object aborts the processing and sends an email, as detailed in the section “Alerting email recipients of processing errors” on page 44. Assume that all fields are complete.

3. The UCCnet_ItemSync collaboration object removes the item from the PROCESSED_GTIN table. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.
4. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to identify the ItemDelist transaction processed. See the section “Using the audit_log table” on page 36 for more information about the audit_log table.
5. The UCCnet_ItemSync collaboration object sends the ItemBasic business object to the ItemCommandRouter collaboration object.
6. The ItemCommandRouter determines that the received business object is a delist.
- 7.

When using a supplier-implemented data source pool (CIN operation):

- a. The ItemCommandRouter sends the ItemBasic business object out the ToCIN_CI port to a CIN_CIP_Dispatcher collaboration object.
- b. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port and maps it to a CatalogueItemNotification_DELIST UCCnetGBO_envelope business object.
- c. It uses the category code from the business object to retrieve the GLNs of any trading partners that subscribe to the category. The GLN information is taken from the GLN subscription file determined by the CIN_DISPATCHER_GLN_FILE property.
- d. For each GLN found in the subscription file, the collaboration object sends a CatalogueItemNotification_DELIST (CIN_DELIST) notification to the AS2 channel connector to be delivered to UCCnet. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.
- e. UCCnet generates an MDN to indicate successful receipt of the CatalogueItemNotification_DELIST message. See “Processing Message Disposition Notifications (MDNs)” on page 32 for details.

When using UCCnet as the data source pool (CIP operation):

- a. The ItemCommandRouter sends the ItemBasic business object out the ToCIN_CI port to the Notify_by_eMail collaboration object.
- b. The Notify_by_eMail collaboration object sends a note to the supplier, indicating that CI_DELIST is unsupported.

As a result of the ItemDelist workflow, the item has been permanently delisted in UCCnet and removed from the PROCESSED_GTIN table.

ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (schema support)

The ItemWithdrawal workflow requests that UCCnet make an item temporarily unavailable to all or selected trading partners. An item might be temporarily removed, for instance, if it is out of season or not in production. It might also be made available only to a specific set of demand-side trading partners as a special order item. (To remove an item from active trading permanently, issue an ItemDelist, as discussed in the section “ItemDelist workflow: making an item permanently unavailable to trading partners (schema support)” on page 22.) The

source of the flow is the withdrawal of an existing item in the ERP source application. This workflow does not result in notifications being sent to demand-side trading partners.

Note: If you are not using schema support, refer to the documentation found in “ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (DTD support)” on page 14

The following description shows how high-level components of the IBM WebSphere Business Integration Collaboration for UCCnet Item Synchronization perform the ItemWithdrawal workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector specific to that ERP. The connector converts item into an application specific business object, and passes it through a map to convert it to an ItemBasic generic business object, and then passes the ItemBasic business object to the UCCnet_ItemSync collaboration object with a Withdraw verb.
2. The UCCnet_ItemSync collaboration object accepts the object and checks the required fields for information as detailed in the section “Checking that item data exists for fields required by UCCnet” on page 34. If all required fields are complete, the collaboration object continues processing it. Otherwise, the collaboration object aborts the processing and sends an email, as detailed in the section “Alerting email recipients of processing errors” on page 44. Assume that all fields are complete.
3. The UCCnet_ItemSync collaboration object changes the Withdrawn column in the PROCESSED_GTIN table to a value of Y. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.
4. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to identify the ItemWithdrawal transaction processed. See the section “Using the audit_log table” on page 36 for more information about the audit_log table.
5. The UCCnet_ItemSync collaboration object sends the ItemBasic business object to the ItemCommandRouter collaboration object.
6. The ItemCommandRouter collaboration object determines from the business object’s verb and the DeleteFlag value that the item is being withdrawn.
- 7.

When using a supplier-implemented data source pool (CIN operation):

- a. The ItemCommandRouter sends the ItemBasic business object out the ToCIN_CI port to a CIN_CIP_Dispatcher collaboration object.
- b. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port and maps it to a CatalogueItemNotification_WITHDRAW UCCnetGBO_envelope business object.
- c. It uses the category code from the business object to retrieve the GLNs of any trading partners that subscribe to the category. The GLN information is taken from the GLN subscription file determined by the CIN_DISPATCHER_GLN_FILE property.
- d. For each GLN found in the subscription file, the collaboration object sends a CatalogueItemNotification_WITHDRAW (CIN_WITHDRAW) notification to the AS2 channel connector to be delivered to UCCnet. See “Sending information from collaboration objects to UCCnet” on page 33 for more details.

- e. UCCnet generates an MDN to indicate successful receipt of the CatalogueItemNotification_WITHDRAW message. See “Processing Message Disposition Notifications (MDNs)” on page 32 for details.

When using UCCnet as the data source pool (CIP operation):

- a. The ItemCommandRouter sends the ItemBasic business object out the ToCIN_CI port to a Notify_by_eMail collaboration object.
- b. The Notify_by_eMail collaboration object receives the business object and sends a note to the supplier saying that the CI_WITHDRAW is unsupported.

As a result of the ItemWithdrawal workflow, the item has been temporarily withdrawn from UCCnet and has been indicated as withdrawn in the PROCESSED_GTIN table.

Maps and data handlers

As the data flows through the workflows, the form of the information changes. Sometimes it is stored in a generic business object, sometimes in an application specific business object, and sometimes, the data is contained in an XML message. As part of the processing, data passes through a variety of transformations. Normally, these transformations are done to prepare the data for the next step in the workflow, whether it be a collaboration object, a connector, or the AS2 channel server.

The following tables detail how the format of the information changes as it moves between the connectors, collaboration objects, and other parts that make up a workflow. The **Transform initiator** and **Time initiated** columns indicate the connector or collaboration object that initiated the data transform, and the point in the workflow that it does this. The **Input** and **Output** columns respectively indicate the format of the data before and after the transformation takes place. The **Transformer** column indicates the map or data handler used to carry out the transformation.

Table 1. Transformers used for DTD Support using iSoft connectivity. The following table indicates how the format of information changes as it moves between parts of a workflow. It also indicates what transformers are used to accomplish these changes. Information for the iSoft Connector controller and iSoft Connector agent is also applicable for the JTextlSoft Connector controller and JTextlSoft Connector agent.

Transform initiator	Time initiated	Input	Output	Transformer
iSoft Connector controller	Sending information to UCCnet	ItemBasic generic business object (GBO)	UCCnetDTD_envelope application specific business object (ASBO)	RouterMap_CwItemBasic_to_UCCnetDTD_envelope map and the transformation maps that it calls: <ul style="list-style-type: none"> CwItemBasic_to_UCCnetDTD_envelope_documentCommand_item CwItemBasic_to_UCCnetDTD_envelope_publishCommand_documentIdentifier
iSoft Connector controller	Sending information to UCCnet	UCCnetGBO_envelope GBO	UCCnetDTD_envelope ASBO	RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope map and the transformation maps that it calls: <ul style="list-style-type: none"> UCCnetGBO_envelope_to_UCCnetDTD_envelope UCCnetGBO_envelope_notification_to_UCCnetDTD_envelope_publishCommand
iSoft Connector controller	Receiving information from UCCnet	UCCnetDTD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetDTD_envelope_to_UCCnetGBO_envelope map
iSoft Connector agent	Receiving information from UCCnet	XML message in UCCnet format	UCCnetDTD_envelope ASBO	IBM WebSphere Business Integration DataHandler for XML
iSoft Connector agent	Sending information to UCCnet	UCCnetDTD_envelope ASBO	XML message in UCCnet format	IBM WebSphere Business Integration DataHandler for XML
SAP connector controller (ERP specific connector in the example workflows)	Receiving item information from SAP application	SAP ASBO	ItemBasic GBO	Sa4CwItemBasic map
SAP Connector controller (ERP specific connector in the example workflows)	Sending item information to SAP application	ItemBasic GBO	SAP ASBO	CwSa4ItemBasic map
JTextRWLconnector agent	Initiating request for UCCnet to return a worklist	XML message in UCCnet format	UCCnetDTD_envelope ASBO	IBM WebSphere Business Integration Data Handler for XML
JTextRWLconnector controller	Initiating request for UCCnet to return a worklist	UCCnetDTD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetDTD_envelope_to_UCCnetGBO_envelope map

Table 2. Transformers used for DTD Support using TPI connectivity. The following table indicates how the format of information changes as it moves between parts of a workflow. It also indicates what transformers are used to accomplish these changes. Information for the TPI Connector controller and TPI Connector agent is also applicable for the JTextTPI Connector controller and JTextTPI Connector agent.

Transform initiator	Time initiated	Input	Output	Transformer
TPI Connector controller	Sending information to UCCnet	ItemBasic generic business object (GBO)	UCCnetTPIIDTD_envelope application specific business object (ASBO)	RouterMap_CwItemBasic_to_UCCnetTPIIDTD_envelope map and the transformation maps that it calls: <ul style="list-style-type: none"> CwItemBasic_to_UCCnetTPIIDTD_envelope_documentCommand_item CwItemBasic_to_UCCnetTPIIDTD_envelope_publishCommand_documentIdentifier
iSoft Connector controller	Sending information to UCCnet	UCCnetGBO_envelope GBO	UCCnetTPIIDTD_envelope ASBO	RouterMap_UCCnetGBO_envelope_to_UCCnetTPIIDTD_envelope map and the transformation maps that it calls: <ul style="list-style-type: none"> UCCnetGBO_envelope_to_UCCnetTPIIDTD_envelope UCCnetGBO_envelope_notification_to_UCCnetTPIIDTD_envelope_publishCommand
TPI Connector controller	Receiving information from UCCnet	UCCnetTPIIDTD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetTPIIDTD_envelope_to_UCCnetGBO_envelope map
TPI Connector agent	Receiving information from UCCnet	XML message in UCCnet format	UCCnetTPIIDTD_envelope ASBO	IBM WebSphere Business Integration DataHandler for XML
TPI Connector agent	Sending information to UCCnet	UCCnetTPIIDTD_envelope ASBO	XML message in UCCnet format	IBM WebSphere Business Integration DataHandler for XML
SAP connector controller (ERP specific connector in the example workflows)	Receiving item information from SAP application	SAP ASBO	ItemBasic GBO	Sa4CwItemBasic map
SAP Connector controller (ERP specific connector in the example workflows)	Sending item information to SAP application	ItemBasic GBO	SAP ASBO	CwSa4ItemBasic map
JTextRWLConnector agent	Initiating request for UCCnet to return a worklist	XML message in UCCnet format	UCCnetTPIIDTD_envelope ASBO	IBM WebSphere Business Integration Data Handler for XML
JTextRWLConnector controller	Initiating request for UCCnet to return a worklist	UCCnetTPIIDTD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetTPIIDTD_envelope_to_UCCnetGBO_envelope map

Table 3. Transformers used for DTD Support using WebSphere Business Integration Connect. The following table indicates how the format of information changes as it moves between parts of a workflow. It also indicates what transformers are used to accomplish these changes. Information for the JMS Connector controller and JMS Connector agent is also applicable for the JTextJMS Connector controller and JTextJMS Connector agent.

Transform initiator	Time initiated	Input	Output	Transformer
JMS Connector controller	Sending information to UCCnet	ItemBasic generic business object (GBO)	UCCnetJMSDITD_envelope application specific business object (ASBO)	RouterMap_CwItemBasic_to_UCCnetJMSDITD_envelope map and the transformation maps that it calls: <ul style="list-style-type: none"> CwItemBasic_to_UCCnetJMSDITD_envelope_documentCommand_item CwItemBasic_to_UCCnetJMSDITD_envelope_publishCommand_documentIdentifier
JMS Connector controller	Sending information to UCCnet	UCCnetGBO_envelope GBO	UCCnetJMSDITD_envelope ASBO	RouterMap_UCCnetGBO_envelope_to_UCCnetJMSDITD_envelope map and the transformation maps that it calls: <ul style="list-style-type: none"> UCCnetGBO_envelope_to_UCCnetJMSDITD_envelope UCCnetGBO_envelope_notification_to_UCCnetJMSDITD_envelope_publishCommand
JMS Connector controller	Receiving information from UCCnet	UCCnetJMSDITD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetJMSDITD_envelope_to_UCCnetGBO_envelope map
JMS Connector agent	Receiving information from UCCnet	XML message in UCCnet format	UCCnetJMSDITD_envelope ASBO	IBM WebSphere Business Integration DataHandler for XML
JMS Connector agent	Sending information to UCCnet	UCCnetJMSDITD_envelope ASBO	XML message in UCCnet format	IBM WebSphere Business Integration DataHandler for XML
SAP connector controller (ERP specific connector in the example workflows)	Receiving item information from SAP application	SAP ASBO	ItemBasic GBO	Sa4CwItemBasic map
SAP Connector controller (ERP specific connector in the example workflows)	Sending item information to SAP application	ItemBasic GBO	SAP ASBO	CwSa4ItemBasic map
JTextRWLConnector agent	Initiating request for UCCnet to return a worklist	XML message in UCCnet format	UCCnetJMSDITD_envelope ASBO	IBM WebSphere Business Integration Data Handler for XML
JTextRWLConnector controller	Initiating request for UCCnet to return a worklist	UCCnetJMSDITD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetJMSDITD_envelope_to_UCCnetGBO_envelope map

Table 4. Transformers used for XSD support using iSoft connectivity. The following table indicates how the format of information changes as it moves between parts of a workflow. It also indicates what transformers are used to accomplish these changes. Information for the iSoft Connector controller and iSoft Connector agent is also applicable for the JTextlSoft Connector controller and JTextlSoft Connector agent.

Transform initiator	Time initiated	Input	Output	Transformer
iSoft Connector controller	Sending information to UCCnet in ItemAdd and ItemChange workflows	ItemBasic generic business object (GBO)	UCCnetXSD_envelope application specific business object (ASBO)	CIN operation CwItemBasic_to_UCCnetXSD_envelope_registerCommand_itemAddChange map
iSoft Connector controller (second instance)	Sending information to UCCnet in ItemPublication and ItemNotification workflows	ItemBasic GBO	UCCnetXSD_envelope ASBO	CIP operation CwItemBasic_to_UCCnetXSD_envelope_registerCommand_CIP_itemAddChange map
iSoft Connector controller	Receiving information from UCCnet	UCCnetXSD_envelope ASBO	UCCnetGBO_envelope GBO	CIP operation only CwItemBasic_to_UCCnetXSD_envelope_publicationCommand_catalogueItem map
iSoft Connector controller	Sending information to UCCnet	UCCnetGBO_envelope GBO	UCCnetXSD_envelope ASBO	UCCnetXSD_envelope_to_UCCnetGBO_envelope map
iSoft Connector agent	Sending information to UCCnet	UCCnetXSD_envelope ASBO	XML message in UCCnet format	IBM WebSphere Business Integration DataHandler for XML
iSoft Connector agent	Receiving information from UCCnet	XML message in UCCnet format	UCCnetXSD_envelope ASBO	IBM WebSphere Business Integration DataHandler for XML
SAP connector controller (ERP specific connector in the example workflows)	Receiving item information from SAP application	SAP ASBO	ItemBasic GBO	Sa4CwItemBasic map
SAP Connector controller (ERP specific connector in the example workflows)	Sending item information to SAP application	ItemBasic GBO	SAP ASBO	CwSa4ItemBasic map
JTextRWLConnector Connector agent	Initiating request for UCCnet to return a workflow	XML message in UCCnet format	UCCnetXSD_envelope ASBO	IBM WebSphere Business Integration DataHandler for XML
JTextRWL Connector controller	Initiating request for UCCnet to return a workflow	UCCnetXSD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetXSD_envelope_to_UCCnetGBO_envelope map
CIN_CIP_Dispatcher collaboration object	Sending CIN or CIP requests to UCCnet	ItemBasic GBO	UCCnetGBO_envelope GBO	CIN operation CwItemBasic_to_UCCnetGBO_envelope_notifyCommand_catalogueItem map
				CIP operation CwItemBasic_to_UCCnetGBO_env_publicationCommand_catalogueItemPublication map

Table 5. Transformers used for XSD support using TPI connectivity. The following table indicates how the format of information changes as it moves between parts of a workflow. It also indicates what transformers are used to accomplish these changes. Information for the TPI Connector controller and TPI Connector agent is also applicable for the JTextTPI Connector controller and JTextTPI Connector agent.

Transform initiator	Time initiated	Input	Output	Transformer
TPI Connector controller	Sending information to UCCnet in ItemAdd and ItemChange workflows	ItemBasic generic business object (GBO)	UCCnetTPIXSD_envelope application specific business object (ASBO)	CIN operation CwItemBasic_to_UCCnetTPIXSD_envelope_registerCommand_itemAddChange map CIP operation CwItemBasic_to_UCCnetTPIXSD_envelope_registerCommand_CIP_itemAddChange map
TPI Connector controller (second instance)	Sending information to UCCnet in ItemPublication and ItemNotification workflows	ItemBasic GBO	UCCnetTPIXSD_envelope ASBO	CIP operation only CwItemBasic_to_UCCnetTPIXSD_envelope_publicationCommand_catalogueItem map
TPI Connector controller	Receiving information from UCCnet	UCCnetTPIXSD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetTPIXSD_envelope_to_UCCnetGBO_envelope map
TPI Connector controller	Receiving information from UCCnet	UCCnetGBO_envelope GBO	UCCnetTPIXSD_envelope ASBO	UCCnetGBO_envelope_to_UCCnetTPIXSD_envelope map
TPI Connector agent	Sending information to UCCnet	UCCnetTPIXSD_envelope ASBO	XML message in UCCnet format	IBM WebSphere Business Integration DataHandler for XML
TPI Connector agent	Receiving information from UCCnet	XML message in UCCnet format	UCCnetTPIXSD_envelope ASBO	IBM WebSphere Business Integration DataHandler for XML
SAP connector controller (ERP specific connector in the example workflows)	Receiving item information from SAP application	SAP ASBO	ItemBasic GBO	Sa4CwItemBasic map
SAP Connector controller (ERP specific connector in the example workflows)	Sending item information to SAP application	ItemBasic GBO	SAP ASBO	CwSa4ItemBasic map
JTextRWLConnector Connector agent	Initiating request for UCCnet to return a worklist	XML message in UCCnet format	UCCnetTPIXSD_envelope ASBO	IBM WebSphere Business Integration Data Handler for XML
JTextRWL Connector controller	Initiating request for UCCnet to return a worklist	UCCnetTPIXSD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetTPIXSD_envelope_to_UCCnetGBO_envelope map
CIN_CIP_Dispatcher collaboration object	Sending CIN or CIP requests to UCCnet	ItemBasic GBO	UCCnetGBO_envelope GBO	CIN operation CwItemBasic_to_UCCnetGBO_envelope_notifyCommand_catalogueItem map CIP operation CwItemBasic_to_UCCnetGBO_env_publicationCommand_catalogueItemPublication map

Table 6. Transformers used for XSD support using WebSphere Business Integration Connect. The following table indicates how the format of information changes as it moves between parts of a workflow. It also indicates what transformers are used to accomplish these changes. Information for the JMS Connector controller and JMS Connector agent is also applicable for the JTextJMS Connector controller and JTextJMS Connector agent.

Transform initiator	Time initiated	Input	Output	Transformer
JMS Connector controller	Sending information to UCCnet in ItemAdd and ItemChange workflows	ItemBasic_generic business object (GBO)	UCCnetJMSXSD_envelope application specific business object (ASBO)	CIN operation CwItemBasic_to_UCCnetJMSXSD_envelope_registerCommand_itemAddChange map
JMS Connector controller (second instance)	Sending information to UCCnet in ItemPublication and ItemNotification workflows	ItemBasic GBO	UCCnetJMSXSD_envelope ASBO	CIP operation CwItemBasic_to_UCCnetJMSXSD_envelope_registerCommand_CIP_itemAddChange map
TPI Connector controller	Receiving information from UCCnet	UCCnetJMSXSD_envelope ASBO	UCCnetGBO_envelope GBO	CIP operation only CwItemBasic_to_UCCnetJMSXSD_envelope_publicationCommand_catalogueItem map
JMS Connector controller	Receiving information from UCCnet	UCCnetGBO_envelope GBO	UCCnetJMSXSD_envelope ASBO	UCCnetJMSXSD_envelope_to_UCCnetGBO_envelope map
JMS Connector agent	Sending information to UCCnet	UCCnetJMSXSD_envelope ASBO	XML message in UCCnet format	IBM WebSphere Business Integration DataHandler for XML
JMS Connector agent	Receiving information from UCCnet	XML message in UCCnet format	UCCnetJMSXSD_envelope ASBO	IBM WebSphere Business Integration DataHandler for XML
SAP connector controller (ERP specific connector in the example workflows)	Receiving item information from SAP application	SAP ASBO	ItemBasic GBO	Sa4CwItemBasic map
SAP Connector controller (ERP specific connector in the example workflows)	Sending item information to SAP application	ItemBasic GBO	SAP ASBO	CwSa4ItemBasic map
JTextRWLConnector Connector agent	Initiating request for UCCnet to return a workload	XML message in UCCnet format	UCCnetJMSXSD_envelope ASBO	IBM WebSphere Business Integration Data Handler for XML
JTextRWL Connector controller	Initiating request for UCCnet to return a workload	UCCnetJMSXSD_envelope ASBO	UCCnetGBO_envelope GBO	UCCnetJMSXSD_envelope_to_UCCnetGBO_envelope map
CIN_CIP_Dispatcher collaboration object	Sending CIN or CIP requests to UCCnet	ItemBasic GBO	UCCnetGBO_envelope GBO	CIN operation CwItemBasic_to_UCCnetGBO_envelope_notifyCommand_catalogueItem map CIP operation CwItemBasic_to_UCCnetGBO_env_publicationCommand_catalogueItemPublication map

Processing Message Disposition Notifications (MDNs)

In many of the workflows, UCCnet generates a Message Disposition Notification to indicate that it has received a message from the supplier. The typical processing of this MDN is as follows:

1. The AS2 channel server delivers the MDN to the UCCnet_processWorklist collaboration object by way of the IBM WebSphere Business Integration Data Handler for XML and the AS2 channel connector. See “Receiving data for a collaboration object from UCCnet” on page 33 for details.
2. The UCCnet_processWorklist collaboration object receives the MDN inside a generic business object and dispatches it to the SIMPLE_RESPONSE subdiagram for further processing.
3. The SIMPLE_RESPONSE subdiagram uses a Notify_by_eMail collaboration object to send email to a predefined list of recipients. See the sections “SIMPLE_RESPONSE subdiagram” on page 43 and “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45 for more information.

Retrieving worklists from UCCnet

In many of the workflows, UCCnet generates, but does not automatically return, a worklist containing a response to a message received from the supplier. In these cases, the supplier must request the worklist from UCCnet.

Retrieval of a worklist from UCCnet happens as follows:

1. The supplier uses a time-triggered process to move a worklist query command to the event directory of the JTextRWLConnector: This movement process is not a part of the solution, and must be customized by the supplier.
2. The JTextRWLConnector polls its event directory for new query commands at user-defined time intervals.
3. When the JTextRWBConnector finds a query command in the event directory, it sends it to the IBM WebSphere Business Integration Data Handler for XML, which converts it to an application specific business object (ASBO).
4. The JTextRWLConnector passes the business object through a map to convert it to a UCCnetGBO_envelope GBO and then passes it to a UCCnet_requestWorklist collaboration object.
5. The UCCnet_requestWorklist collaboration object, in turn sends the business object to the AS2 channel connector for delivery to UCCnet. See “Sending information from collaboration objects to UCCnet” on page 33 for details.

Customizing the process

The DTD_URL and SET_UNIQUE_IDS properties of the UCCnet_requestWorklist collaboration object affect the outgoing XML message in systems using the DTD XML definition type. The DocType line in the XML is set according to the value of the DTD_URL property. If outgoing messages are required to have unique message IDs, the SET_UNIQUE_IDS property must be set to ALL

Both the worklist request XML and the polling interval can be changed. For example, the worklist query command XML message tailored for Authorization Notifications (query type=“NOTIFICATION” with name=“AUTHORIZATION_INFORMATION” and status=“UNREAD”) can be used to request the worklist authorization notification contents. A similar request can be constructed to read any dead letter notifications. As an alternative, all notifications

can be requested. The polling interval is set in the **PollFrequency** attribute of the `JTextRWLConnector` and is in milliseconds.

The `UCCnet_requestWorklist` collaboration supports the notification type="PUBLICATION_INFORMATION" for the topics `PEND_PUBLICATION`, `PRE_AUTHORIZATION`, `AUTHORIZATION`, `REJECT_PUBLICATION`, `DE_AUTHORIZATION`, and in the notifications for `NEW_ITEM_PUBLICATION_REQUEST` and `ITEM_INFORMATION` (`ITEM_ADD`, `ITEM_CHANGE`).

Sending information from collaboration objects to UCCnet

Sending messages from a collaboration object to UCCnet:

In many of the workflows, a collaboration object needs to send information to UCCnet. The following set of events accomplishes this action:

1. The collaboration object sends a generic business object (GBO) to the AS2 channel connector.
2. The connector controller calls the associated map configured for this GBO to convert it into an application specific business object (ASBO).
3. The connector controller passes the ASBO to the connector agent.
4. The connector agent calls the IBM WebSphere Business Integration Data Handler for XML which converts the ASBO into a UCCnet formatted XML message.
5. The AS2 channel connector agent then passes the XML message to the AS2 channel server, which creates the digest, encrypts, and transmits the message to UCCnet.

Receiving data for a collaboration object from UCCnet

Receiving messages back from UCCnet:

In many of the workflows, UCCnet will send a message to the AS2 channel server to be passed on to a collaboration object. The following set of events accomplishes this action:

1. UCCnet sends an XML message to the AS2 channel server.
2. The AS2 channel connector agent retrieves the message from the AS2 channel server.
3. The connector agent calls the IBM WebSphere Business Integration Data Handler for XML to convert the message into an application specific business object (ASBO).
4. The connector agent passes the ASBO to the connector controller.
5. The connector controller calls the associated map configured for this ASBO to convert it into a generic business object (GBO).
6. The AS2 channel connector then passes the GBO to all collaboration objects that subscribe to the GBO.

Checking that item data exists for fields required by UCCnet

UCCnet requires its community of trading partners to provide standardized item data in particular formats to its registry. As a result, UCCnet requires requests for ItemAdd, ItemChange, ItemDelist, and ItemWithdrawal publications to have data provided for certain fields. If the data for the required fields is not present, UCCnet does not process the publications. Data might be missing if the ERP does not require information for these same fields and the ERP user is not aware of the UCCnet requirements.

To help ensure that ItemAdd, ItemChange, ItemDelist, and ItemWithdrawal publications are accepted by UCCnet, when a UCCnet_ItemSync collaboration object accepts an ItemBasic business object, it checks that the following fields that are required by UCCnet to have data contain information (i.e., are not NULL):

- For ItemAdd and ItemChange publications:
 - gtin
 - dimension
 - height
 - volume
 - productHierarchy
 - barCodeId
 - unitOfWgt (if either the grossWeight or netWeight fields contain values)
- For ItemDelist and ItemWithdrawal publications:
 - gtin

Note: The UCCnet_ItemSync collaboration object checks only that the required fields are not NULL. It does not verify that the information within them is in the correct format for UCCnet.

If all required fields are complete, the UCCnet_ItemSync collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an email requesting the missing information to an email address provided in the UCCnet_ItemSync collaboration object's SEND_MAIL_TO configuration property. See the section "Alerting email recipients of processing errors" on page 44 for more information on how email is handled within the solution.

Using the PROCESSED_GTIN table

The PROCESSED_GTIN table is a relational table provided with the Item Synchronization for Suppliers solution. It maintains the complete list of the supplier's items that exist in the UCCnet repository by using each item's tracking ID, or GTIN, as the primary key. This table permits a UCCnet_processWorklist collaboration object to process incoming INITIAL_ITEM_LOAD_REQUEST commands without the need to communicate with the back-end ERP system. See the section "INITIAL_ITEM_LOAD_REQUEST subdiagram" on page 40 for more information on this subdiagram.

The UCCnet_ItemSync and UCCnet_processWorklist collaboration objects both interact with this table. A UCCnet_ItemSync collaboration object updates the table each time it processes an ItemBasic business object. The collaboration object performs this processing only if all of the fields for which UCCnet requires data are complete (see the section "Checking that item data exists for fields required by

UCCnet” on page 34 for more information). The type of processing the collaboration object performs depends on the verb attached to the ItemBasic business object, as follows:

- If the ItemBasic business object has a *Create* verb, the UCCnet_ItemSync collaboration object checks if the item exists in the PROCESSED_GTIN table and processes it, as follows:
 - If the item does not already exist in the PROCESSED_GTIN table, the collaboration object adds an entry for it to the table, and sets the value for the **withdrawn** field for this entry to N.
 - If the item already exists in the table, the collaboration object changes its verb to *Update*.
- If the ItemBasic business object has an *Update* verb, the UCCnet_ItemSync collaboration object checks if the item exists in the PROCESSED_GTIN table and processes it, as follows:
 - If the item exists in the table and the value for its **withdrawn** field is set to N, the collaboration object continues processing it.
 - If the item exists in the table and the value for its **withdrawn** field is set to Y, the collaboration object does the following:
 - Changes the value of the entry’s **withdrawn** field to N.
 - Changes the value of the entry’s **delete** field to U.
 - Changes the business object verb to *UNWITHDRAWN*.
 - Continues processing it.
 - If the item does not exist in the table, the collaboration object changes the business object’s verb to *Create* and adds it to the PROCESSED_GTIN table, setting the entry’s **withdrawn** field to N.
- If the ItemBasic business object has a *Delist* verb, the UCCnet_ItemSync collaboration object removes the item from the PROCESSED_GTIN table.
- If the ItemBasic business object has a *Withdraw* verb, the UCCnet_ItemSync collaboration object locates the item in the PROCESSED_GTIN table and sets the value for the entry’s **withdrawn** field to Y. This action prevents the publication of the item in response to an incoming INITIAL_ITEM_LOAD_REQUEST.

The UCCnet_processWorklist collaboration object reads this table during processing, as follows:

- Its NEW_ITEM_PUBLICATION_REQUEST subdiagram verifies that specific items are in the table.
- Its INITIAL_ITEM_LOAD_REQUEST subdiagram generates publication messages for all items in the table with a withdrawn value of N.

The UCCnet_ItemSync collaboration object connects to the database through its GtinDB_USER, GtinDB_PASSWORD, JDBC_DRIVER, and JDBC_URL configuration properties; the UCCnet_processWorklist collaboration object, through its DB_USER, DB_PASSWORD, JDBC_DRIVER, and JDBC_URL configuration properties. See Collaboration for UCCnet Item Synchronization for detailed information on these properties.

Connection information for the PROCESSED_GTIN table is configured as part of the UCCnet_ItemSync collaboration object setup. After the relationships have been deployed, the table itself is created by running the supplied InitializeRelationshipTables.sql file for the database type (i.e., DB2®, Oracle, or Microsoft® SQL Server). See the Installation guide for installation instructions.

Using the audit_log table

The audit_log table is provided with the Item Synchronization for Suppliers solution. It is used to track the events associated with UCCnet activities to support complete end-to-end auditing. This audit support provides irrefutable documentation that transmissions have occurred between trading partners. It also provides a profile of which trading partners are participating in the trading community and with which products that participation is associated.

The audit service is composed of four components:

- The audit_log table receives log entries from each participant component in the solution. The following table lists the fields that can appear in an audit_log entry. An example value is given for each field.

Table 7. Audit_log table fields

Field	Example value
LOG_SEQ_NO	1
LOG_SOURCE_NAME	UCCnet2
GLN_CODE	NA
SOURCE_SYSTEM	
PRODUCT_ID	2050000000454
VERB_NAME	Create
TRANS_ID	SAPConnector_1015606877187_1
TRANS_TYPE	UCCnet_processWorklist
TRANS_STATUS	ITEM_ADD_CHANGE
MSG_FILEPATH_TEXT	C:\IBM\WebSphereICS\UCCnet-1051628537493.bo
LOG_DTTM	May 5, 2003 1:44:56 PM

- A logging framework that is utilized by IBM WebSphere Business Integration Collaborations and Adapters to log critical events to the audit_log table.
- An audit log merge process that periodically sweeps the TPI server log for new entries, which are added to the unified audit_log table. The iSoft Peer-to-Peer Agent logs are not swept for new entries.
- A report generation facility to support data analysis and visualization.

UCCnet_ItemSync and UCCnet_processWorklist collaboration objects impact the audit_log table. Any time an item is added to, updated or delisted within, or withdrawn from the ERP, and an ItemBasic business object is subsequently passed to a UCCnet_ItemSync collaboration object, the collaboration object records an entry in the audit_log table detailing the event. The following subdiagrams of the UCCnet_processWorklist collaboration object also record entries in the audit_log table during processing:

- AUTHORIZATION_RESPONSES
- CATALOGUE_ITEM_CONFIRMATION
- INITIAL_ITEM_LOAD_REQUEST
- ITEM_ADD_CHANGE
- NEW_ITEM_PUBLICATION_REQUEST
- RCIR_RESPONSE

The following sections have more information on how these subdiagrams operate:

- “AUTHORIZATION_RESPONSES subdiagram” on page 39
- “CATALOGUE_ITEM_CONFIRMATION subdiagram” on page 40
- “INITIAL_ITEM_LOAD_REQUEST subdiagram” on page 40
- “ITEM_ADD_CHANGE subdiagram” on page 41
- “NEW_ITEM_PUBLICATION_REQUEST subdiagram” on page 41
- “RCIR_RESPONSE subdiagram” on page 42

Each audit entry is associated with the value listed for the collaboration object’s SUPPLIER_NAME attribute.

Connection from the UCCnet_ItemSync and UCCnet_processWorklist collaboration objects to the audit_log table is provided by the IBM JDBC Driver for DB2, Oracle, or Microsoft SQL Server. Table creation is performed via running the supplied audit_log.sql file for the database type (i.e., DB2, Oracle, or Microsoft SQL Server). See the Installation guide for installation instructions.

Using the trading_partner table

The trading_partner table, or GLN table, is a relational table provided with the Item Synchronization for Suppliers solution. It maintains the complete list of trading partners by using the Global Location Number (GLN) of each as the key.

The NEW_ITEM_PUBLICATION_REQUEST subdiagram of a UCCnet_processWorklist collaboration object checks that a new item or updated item information to be published is supplied by a trading partner from this table. It also verifies that the demand-side trading partners to whom notification will be sent are in the table. The INITIAL_ITEM_LOAD_REQUEST subdiagram of a UCCnet_processWorklist collaboration object checks the trading_partner table to verify that the demand-side trading partner requesting the INITIAL_ITEM_LOAD_REQUEST exists. The ITEM_ADD_CHANGE subdiagram of a UCCnet_processWorklist collaboration object utilizes maps that read from the trading_partner table.

Connection information for the trading_partner table is configured as part of the UCCnet_processWorklist collaboration object.

The table itself is created when the TPTable relationship is deployed (and the schema is created). The InitializeRelationshipTables scripts (InitializeRelationshipTables.sql and InitializeRelationshipTablesForXSD.sql) make alterations to the table then make alteration to the table. These changes include the addition of the following columns to the table:

The following columns are added to the table via running the supplied InitializeRelationshipTables.sql file for the database type (i.e., DB2, Oracle, or Microsoft SQL Server):

- gln_code
- trading_partner_name
- trading_partner_contact
- trading_partner_group
- trading_partner_type
- initial_load_flag

After the columns are created, you must manually populate it with the correct information. See the Installation guide for installation instructions.

Using subdiagrams

All messages initiated from UCCnet are in UCCnet XML format. Since the UCCnet XML format's top-level tag (<envelope>) is the same for all messages, a component is needed to distinguish among the various notification and response XML messages and return different business objects for them. An object based on the UCCnet_processWorklist collaboration template performs this task.

A UCCnet_processWorklist collaboration object is instantiated when the AS2 channel connector forwards a UCCnetGBO_envelope business object to it. See "Receiving data for a collaboration object from UCCnet" on page 33 for details of how this business object is created.

The UCCnet_processWorklist collaboration object parses the UCCnetGBO_envelope business object, creating a separate UCCnetGBO_envelope business object for each notification or response. The collaboration object routes each business object representing a single notification to the appropriate subdiagram. Each subdiagram handles a particular set of notification or response messages, as follows:

- AUTHORIZATION_RESPONSES subdiagram — Handles notifications for the topics AUTHORIZE, DE_AUTHORIZATION, PEND_PUBLICATION, PRE_AUTHORIZATION, and REJECT_PUBLICATION. See the section "AUTHORIZATION_RESPONSES subdiagram" on page 39 for more information on this subdiagram.
- CATALOGUE_ITEM_CONFIRMATION — Handles the process of the CATALOGUE_ITEM_CONFIRMATION responses, received by the UCCnet_processWorklist collaboration object. See "CATALOGUE_ITEM_CONFIRMATION subdiagram" on page 40 for more information.
- CATEGORY_ADD_CHANGE subdiagram — Handles notifications for the CATEGORY_ADD and CATEGORY_CHANGE topics. See the section "CATEGORY_ADD_CHANGE subdiagram" on page 39 for more information on this subdiagram.
- CIN_RESPONSE subdiagram — Handles incoming messages that are recognized as CIN_RESPONSE messages. See the section "CIN_RESPONSE subdiagram" on page 40 for more information on this subdiagram.
- DEAD_LETTER_PUB_RECEIPT subdiagram — Handles notifications associated with the single notification topic DEAD_LETTER_PUB_RECEIPT. See the section "DEAD_LETTER_PUB_RECEIPT subdiagram" on page 40 for more information on this subdiagram.
- INITIAL_ITEM_LOAD_REQUEST subdiagram — Handles notifications associated with the single notification topic INITIAL_ITEM_LOAD_REQUEST. See the section "INITIAL_ITEM_LOAD_REQUEST subdiagram" on page 40 for more information on this subdiagram.
- ITEM_ADD_CHANGE subdiagram — Handles notifications for the ITEM_ADD and ITEM_CHANGE topics. See the section "ITEM_ADD_CHANGE subdiagram" on page 41 for more information on this subdiagram.
- NEW_ITEM_PUBLICATION_REQUEST subdiagram — Handles notifications associated with the single notification topic NEW_ITEM_PUBLICATION_REQUEST. See the section "NEW_ITEM_PUBLICATION_REQUEST subdiagram" on page 41 for more information on this subdiagram.

- PUBLICATION_COMMAND_RESPONSE subdiagram — Handles incoming messages that are recognized as CI_RESPONSE or CIP_RESPONSE messages. See the section “PUBLICATION_COMMAND_RESPONSE subdiagram” on page 42 for more information on this subdiagram
- RCIR_RESPONSE — Handles notifications for the ADD and CHANGE topics. See the section “RCIR_RESPONSE subdiagram” on page 42 for more information.
- RCIR_QUERY_RESPONISE — Handles incoming messages that are recognized as RCIR_QUERY_RESPONSE messages. See the section “RCIR_QUERY_RESPONSE subdiagram” on page 43 for more information.
- SIMPLE_RESPONSE subdiagram — Handles immediate responses to commands such as MDNs from UCCnet. See the section “SIMPLE_RESPONSE subdiagram” on page 43 for more information on this subdiagram.
- UNKNOWN_MESSAGES subdiagram — Handles incoming messages not recognized as supported. See the section “UNKNOWN_MESSAGES subdiagram” on page 43 for more information on this subdiagram.
- UNKNOWN_RESPONSE subdiagram — Handles incoming messages that are recognized as notification messages, but are not supported. See the section “UNKNOWN_RESPONSE subdiagram” on page 44 for more information on this subdiagram.

AUTHORIZATION_RESPONSES subdiagram

This subdiagram handles notifications for the following topics: AUTHORIZE, DE_AUTHORIZATION, PEND_PUBLICATION, PRE_AUTHORIZATION, and REJECT_PUBLICATION. UCCnet generates these notifications as a result of authorization actions taken by demand-side trading partners, which are forwarded to UCCnet. Typically they are issued in response to an ItemPublicationAdd issued by the supply-side trading partner, but they can be issued at anytime. The subdiagram logic does the following:

1. Instantiates a collaboration object based on the Notify_by_eMail collaboration template, which sends an email to selected recipients. The email message, subject, and recipients are configured in this collaboration object’s EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. See the section “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45 for more information.
2. Logs the event in the audit_log table. See “Using the audit_log table” on page 36 for more information on the audit_log table.

CATEGORY_ADD_CHANGE subdiagram

This subdiagram handles notifications for the CATEGORY_ADD and CATEGORY_CHANGE topics. These notifications are sent by UCCnet when a request is made by the supply-side trading partner to add or change a category to better classify or organize the items in its available inventory. The subdiagram logic sends an email containing the category maintenance information to selected recipients by instantiating a collaboration object based on the Notify_by_eMail collaboration template.. The email message, subject, and recipients are configured in this collaboration object’s EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. See the section “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45 for more information. There is no follow-on flow.

CATALOGUE_ITEM_CONFIRMATION subdiagram

This subdiagram handles the process of the CATALOGUE_ITEM_CONFIRMATION responses, received by the UCCnet_processWorklist collaboration object. UCCnet generates these responses as a result of the authorization actions taken by demand-side trading partners. The responses can have one of the following states:

- SYNCHRONISED
- ACCEPTED
- REVIEW
- REJECTED

The responses are normally generated in answer to a request generated by the supply-side trading partner as part of the CatalogueItemNotification_Change and CatalogueItemNotification_Add workflows. However, the responses can be issued at any time. This subdiagram carries out the following logic:

- It instantiates a collaboration object based on the Notify_by_eMail collaboration template. This collaboration object sends an email to a set of defined recipients. The message, subject, and recipient list are defined by the collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. See the section "Sending email through UCCnet_processWorklist collaboration object subdiagrams" on page 45 for more information.
- It logs the event in the audit_log table. See the section "Using the audit_log table" on page 36 for more information.

CIN_RESPONSE subdiagram

This subdiagram handles incoming CIN_Response messages. The subdiagram logic instantiates a collaboration object based on the Notify_by_eMail collaboration template, which sends an email to selected recipients. The email message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. See the section "Sending email through UCCnet_processWorklist collaboration object subdiagrams" on page 45 for more information.

DEAD_LETTER_PUB_RECEIPT subdiagram

This subdiagram handles notifications for the DEAD_LETTER_PUB_RECEIPT topic. These notifications result from a supplier request for which a target demand-side trading partner has not subscribed. The subdiagram logic instantiates a collaboration object based on the Notify_by_eMail collaboration template, which sends an email to selected recipients. The email message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. See the section "Sending email through UCCnet_processWorklist collaboration object subdiagrams" on page 45 for more information.

INITIAL_ITEM_LOAD_REQUEST subdiagram

This subdiagram handles notifications associated with the single notification topic INITIAL_ITEM_LOAD_REQUEST. This notification is generated as a result of a demand-side trading partner requesting through UCCnet to initiate synchronizing all the items currently traded with a given supply-side trading partner. The demand-side partner's request produces a notification in the worklist of the supply-side trading partner.

The logic in this subdiagram does the following:

1. Writes a record to the `audit_log` table indicating receipt of the `INITIAL_ITEM_LOAD_REQUEST`. See the section “Using the `audit_log` table” on page 36 for more information on the `audit_log` table.
2. Checks the `trading_partner` table to verify that the GLN requesting the `INITIAL_ITEM_LOAD_REQUEST` exists. See the section “Using the `trading_partner` table” on page 37 for more information on the `trading_partner` table.
3. Sends the business object to UCCnet over the `INITIAL_ITEM_LOAD_REQUEST` port via the AS2 channel connector.

The follow-up workflow is that of an `ItemPublicationAdd` subflow 1 targeted to the trading partner who initiated the flow, as detailed in the section “`ItemPublicationAdd` subflow 1: making a new item available to trading partners” on page 6. The `PROCESSED_GTIN` table provides the list of GTINs for the `ItemPublicationAdd`, and the incoming message provides the trading partner’s GLN. There are no external business process steps for this flow.

ITEM_ADD_CHANGE subdiagram

This subdiagram handles notifications for the `ITEM_ADD` and `ITEM_CHANGE` topics. These notifications are sent by UCCnet to indicate completion of a particular item synchronization request, such as one initiated by an `ItemAdd` or `ItemChange` workflow. The subdiagram logic does the following:

1. Receives the `UCCnetGBO_envelope` business object.
2. Configures it so that the correct maps will be used by the AS2 channel connector.
3. Sends the `UCCnetGBO_envelope` over the `ITEM_ADD_CHANGE` port to the AS2 channel connector where it is mapped to an `ItemPublicationAdd` or `ItemPublicationChange` request.
4. Logs the notification in the `audit_log` table. See the section “Using the `audit_log` table” on page 36 for more information on the `audit_log` table.

The follow-up workflow is that of the first subflow of either the `ItemPublicationAdd` or `ItemPublicationChange` workflow, as detailed in the sections “`ItemPublicationAdd` subflow 1: making a new item available to trading partners” on page 6 and “`ItemPublicationChange` subflow 1: making updated item information available to trading partners” on page 10.

NEW_ITEM_PUBLICATION_REQUEST subdiagram

This subdiagram handles notifications associated with the single notification topic `NEW_ITEM_PUBLICATION_REQUEST`. This notification is generated as a result of the following:

- A new item being added to the source ERP.
- Item data being updated in the source ERP.
- A demand-side trading partner requesting through UCCnet that a supply-side trading partner publish a specific item (GTIN) or items to it so it can synchronize them. The demand-side partner’s request produces a notification in the worklist of the supply-side trading partner. This notification has the `type="NEW_ITEM_PUBLICATION_REQUEST"`.

The logic in this subdiagram does the following:

1. Verifies that the GTIN value associated with the item is in the PROCESSED_GTIN table and that the item is not withdrawn. See the section “Using the PROCESSED_GTIN table” on page 34 for more information on the PROCESSED_GTIN table.
2. Checks that the new item or new item information to be published is supplied by a trading partner listed in the trading_partner table and verifies that the demand-side trading partners to whom notification will be sent are also in this table. See the section “Using the trading_partner table” on page 37 for more information on the trading_partner table.
3. Configures the business object so that the correct maps will be used by the AS2 channel connector.
4. Logs the notification in the audit_log table. See the section “Using the audit_log table” on page 36 for more information on the audit_log table.

The follow-up workflow is that of the second subflow of either the ItemPublicationAdd or ItemPublicationChange workflow, as detailed in the sections “ItemPublicationAdd subflow 2: processing trading partners’ responses to a new item” on page 7 and “ItemPublicationChange subflow 2: processing trading partners’ responses to updated item information” on page 11.

PUBLICATION_COMMAND_RESPONSE subdiagram

This subdiagram handles incoming CI response and CIP response messages as follows:

1. It writes a record to the audit_log table indicating receipt of the CI or CIP response message. See the section “Using the audit_log table” on page 36 for more information on the audit_log table. .
2. If the received message is a CIP_RESPONSE, then it sends the UCCnetGBO_envelope through the CIP_RESPONSE port to the a Notify_by_eMail collaboration object to notify the supplier, and then exits.
3. Otherwise, it creates an empty ItemBasic business object and sends it to the DestinationAppRetrieve port with a retrieve verb. This actions retrieves the ItemBasic business that initiated the corresponding RCIR command originally sent to UCCnet.
4. If the retrieved ItemBasic business object indicates a CHANGE, the subdiagram sends a UCCnetGBO_envelope business object through CI_RESPONSE port to a Notify_by_eMail collaboration object to notify the supplier, and then exits.
5. If the retrieved ItemBasic BO indicates an ADD, the subdiagram sends the business object out the PUBLICATION_CMD_RESPONSE port to an instance of the CIN_CIP_Dispatcher collaboration object to initiate sending of CIPs.

RCIR_RESPONSE subdiagram

This subdiagram handles notifications for the ADD and CHANGE topics. These notifications are sent by UCCnet when a request is made by the supply-side trading partner to add or change an item.

The subdiagram logic first records the occurrence of the RCIR_RESPONSE message in the audit_log. Then, it builds a skeleton ItemBasic business object, defining only the UPCEANCODE and ITEM_DOMAIN attributes. It next sends this ItemBasic business object out through the DestinationAppRetrieve port so that the user can respond with a completed ItemBasic business object. In a production environment, an additional process is required to retrieve the fully defined ItemBasic business object using the UPCEANCODE and ITEM_DOMAIN fields, and to return it to the DestinationAppRetrieve port.

Once the fully defined ItemBasic business object has been returned:

For CIN operation

The RCIR_RESPONSE subdiagram sends it out through the RCIR_RESPONSE port to the CIN_CIP_Dispatcher collaboration object. This action triggers the CIN_CIP_Dispatcher collaboration object to generate CATALOGUE_ITEM_NOTIFICATION messages and send one to each subscribed demand-side trading partner.

Note: CATALOGUE_ITEM_NOTIFICATION is a general term. The workflow explanations refer to either a CatalogueItemNotificaton_ADD message, or a CatalogueItemNotification_CHANGE as specific instances of a CATALOGUE_ITEM_NOTIFICATION message.

For CIP operation

The RCIR_RESPONSE subdiagram sends it out through the RCIR_RESPONSE port to an AS2 channel connector, which maps it to a Catalogue Item message and sends it to UCCnet.

The follow-up workflow is the CatalogueItemNotification_Add or CatalogueItemNotification_Change workflow, as detailed in the sections “CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses” on page 16 and “CatalogueItemNotification_Change and CatalogueItemPublication_Change workflow: making updated item information available to trading partners and processing their responses” on page 20. Ensure that the schema delist and withdrawal workflows do not use this subdiagram.

RCIR_QUERY_RESPONSE subdiagram

This subdiagram handles incoming RCIR Query response messages. The subdiagram logic instantiates a collaboration object based on the Notify_by_eMail collaboration template, which sends an email to selected recipients with the contents of the received message. The email message, subject, and recipients are configured in this collaboration object’s EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. See the section “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45 for more information.

SIMPLE_RESPONSE subdiagram

This subdiagram handles immediate responses to commands such as MDNs from UCCnet. The subdiagram logic instantiates a collaboration object based on the Notify_by_eMail collaboration template, which sends an email to selected recipients. The email message, subject, and recipients are configured in this collaboration object’s EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. See the section “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45 for more information.

UNKNOWN_MESSAGES subdiagram

This subdiagram handles incoming messages not recognized as supported. The subdiagram logic instantiates a collaboration object based on the Notify_by_eMail collaboration template called, which sends an email to selected recipients. The email message, subject, and recipients are configured in this collaboration object’s EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS

configuration properties, respectively. See the section “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45 for more information.

UNKNOWN_RESPONSE subdiagram

This subdiagram handles incoming messages that are recognized as notification messages, but are not supported. The subdiagram logic instantiates a collaboration object based on the Notify_by_eMail collaboration template, which sends an email to selected recipients. The email message, subject, and recipients are configured in this collaboration object’s EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. See the section “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45 for more information.

Sending email

UCCnet_ItemSync and UCCnet_processWorklist collaboration objects can be configured to send email to alert when processing errors occur. A UCCnet_processWorklist collaboration object can also instantiate collaboration objects based on the Notify_by_eMail collaboration template to respond by email to configured recipients when specific processing circumstances occur.

For more information on these topics, see the following sections:

- “Alerting email recipients of processing errors”
- “Sending email through UCCnet_processWorklist collaboration object subdiagrams” on page 45

Alerting email recipients of processing errors

UCCnet_ItemSync and UCCnet_processWorklist collaboration objects can be configured to send email to alert recipients when processing errors occur.

UCCnet_ItemSync collaboration object

This collaboration object uses two configuration properties to control whether email is sent and to identify the mail recipients.

- SEND_EMAIL — This property controls whether email is sent to the email address specified in the SEND_EMAIL_TO configuration property. Set the property value to all to send email or to none to not send email. If the value is left empty, no email is sent even if recipients exist in the SEND_EMAIL_TO property.
- SEND_EMAIL_TO — This property defines the email addresses to which error messages are sent. Multiple addresses can be provided in a comma-delimited list. This property must be configured by the user.

UCCnet_processWorklist collaboration object

This collaboration object uses one configuration property to control whether email is sent and to identify the mail recipients. The SEND_EMAIL_TO property defines the email addresses to which error messages are sent. Multiple addresses can be provided in a comma-delimited list. If a value exists for this property, the collaboration object sends email. If the property is left blank, the object does not send email. This property must be configured by the user.

Sending email through UCCnet_processWorklist collaboration object subdiagrams

A UCCnet_processWorklist collaboration object contains several subdiagrams that respond to specific types of workflow processing. The following subdiagrams include functionality that sends an email to a set of configured addresses by instantiating a collaboration object based on the Notify_by_eMail collaboration template:

- CATEGORY_ADD_CHANGE subdiagram
- AUTHORIZATION_RESPONSES subdiagram
- DEAD_LETTER_PUB_RECEIPT subdiagram
- CATALOGUE_ITEM_CONFIRMATION subdiagram
- SIMPLE_RESPONSE subdiagram
- UNKNOWN_MESSAGES subdiagram
- UNKNOWN_RESPONSE subdiagram

Each of these subdiagrams instantiates an object based on the Notify_by_eMail collaboration template, which can be configured to contain the email message, subject, and recipients specific to its processing situation through its EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties, respectively. These properties can also contain the names of files, which permits messages, subjects, and recipients to be shared among multiple collaboration objects. Also, more than one recipient can be specified to receive email through use of a comma-delimited list. Plus, email message and subject text can be constants that contain variables. The Notify_by_eMail collaboration object substitutes data from the business object into these variables dynamically. See the following sections for more information on these features:

- “Specifying message text, subjects, and recipients in external files”
- “Specifying changing individual or multiple message recipients” on page 46
- “Using substitution variables in message and subject text” on page 46

The value of the AUTO_RESPOND property of the UCCnet_processWorklist collaboration object determines whether email is sent. The value of this collaboration object’s DTD_URL property sets the DTD line in the XML in any outgoing XML message.

Specifying message text, subjects, and recipients in external files

A Notify_by_eMail collaboration object allows the contents of its properties that specify email message text, subject text, and recipients to contain the names of files. These files contain the actual email message text, subject text, and addresses, and can be easily modified without modifying the using collaboration objects. This feature permits messages, subjects, and recipients to be shared among multiple collaboration objects. A solution’s messages, subjects, and recipients can all be contained in one easily modifiable directory.

A Notify_by_eMail collaboration object uses the following configuration properties to identify the email message text, subject text, and recipients:

EMAIL_MESSAGE

Identifies the message text.

EMAIL_SUBJECT

Identifies the subject text.

EMAIL_NOTIFICATION_RCPTS

Identifies the recipient or list of recipients.

The collaboration object distinguishes whether the content of a property is an actual value or filename based on whether the value is prefixed by the character @. If the value of the property is prefixed with the character @, the Notify_by_eMail collaboration object interprets the rest of the value as a filename. The collaboration object reads the value of the file into a String variable in preparation for further processing. Files must be identified by their fully qualified names.

For instance, if the filename containing the email recipients is `c:\Email_Files\CategoryManagerRole.txt`, set the value of the EMAIL_NOTIFICATION_RCPTS property, as follows:

```
@c:\Email_Files\CategoryManagerRole.txt
```

If the value of a property does not start with the character @, the Notify_by_eMail collaboration object obtains the email value directly from the attribute.

Specifying changing individual or multiple message recipients

A Notify_by_eMail collaboration object allows all email messages to be routed to an administrator or to a specific role in an organization (like a Category Manager), without the need to maintain the email recipient's fully qualified email address in every collaboration object that might send email. By placing the email address in an external file, if the address changes, the file can be modified without having to reconfigure the using collaboration objects. More than one recipient can be specified to receive the email through use of a comma-delimited list. The comma-delimited list can be specified in the business object attribute or in the external file pointed to by the attribute.

Using substitution variables in message and subject text

Email message and subject text can be constants that contain variables. A collaboration object based on the Notify_by_eMail template substitutes data from the business object into these variables dynamically. Variables to be substituted must be enclosed in the prefix characters `${` and the suffix character `}`. As a result, the substitution variables in the email message and subject text must appear as:

```
${variable_name}
```

Note: These characters might have to be changed to meet National Language requirements.

The supported values for *variable_name*, along with the values that the collaboration object actually inserts in the text, are as follows:

getRoot

Substitutes the entire triggering business object.

getDate

Substitutes the current date and time.

getName

Substitutes the name of the triggering business object.

getVerb

Substitutes the verb of the triggering business object.

Any attribute name

Substitutes the value of the named attribute from the triggering business object.

If the value for *variable_name* does not match one of the specific values above, the collaboration object interprets it as the name of a business object attribute. For instance, in the following sample message:

```

UCCnet_processWorklist_AUTHORIZATION_RESPONSES.mail:      \
Date:  ${getDate}                                         \
BusinessObject:  ${getName}.${getVerb}                    \
Topic:                                                    \
${ROOT.body[0].response.acknowledge.acknowledgement.     \
subdocumentValid[0].subdocumentValid[0]resultList[0].    \
notification.topic}                                       \
GLN:                                                       \
${ROOT.body[0].response.acknowledge.acknowledgement.     \
subdocumentValid[0].subdocumentValid[0].resultList[0].   \
notification.notificationDetail.transactionInformation.   \
entityIdentification.globalLocationNumber.gln}           \
GTIN:                                                      \
${TLO.body.body_Wrapper1[0].response.acknowledge.        \
acknowledgement.subdocumentValid[0].subdocumentValid[0]. \
resultList.resultList_Wrapper1[0].notification.         \
notificationDetail.authorizationNotification.publication. \
item.itemInformation.globalTradeItemNumber.gtin}         \

${getRoot}

```

the following variables are filled in automatically during the generation of the message, as follows:

- `${getDate}`, with the current date and time.
- `${getName}`, with the name of the triggering business object.
- `${getVerb}`, with the verb of the triggering business object.
- All variables beginning with `${ROOT.body[0]. . .}`, with the values for those attributes.
- `${getRoot}` with the entire triggering business object.

Logging

If `UCCnet_ItemSync`, `UCCnet_requestWorklist`, `UCCnet_processWorklist`, and `Notify_by_eMail` collaboration objects encounter error situations during any stage of processing, they do the following:

- Log the error in the configured log destination.
- Return the object to the calling collaboration object through the *From* port.

Note: For error logging to occur, tracing must be enabled. Also, use separate files for tracing and logging. Use logging files to maintain persistent records of processed data. Use tracing files to diagnose problems and to show the flow of an item through the Item Synchronization for Suppliers solution. The Log Viewer tool has log and trace file filters that enable users to view the log or trace records for a particular business object or collaboration object.

Tracing

All collaboration objects based on collaboration templates included in the Item Synchronization for Suppliers solution provide tracing capabilities to record logical flows and data processed. Users can enable tracing for a particular collaboration object by selecting the collaboration object in the System Manager, displaying its properties, and, on the **Collaboration General Properties** tab, selecting a trace level greater than 0 from the **System trace level** field.

Enable tracing for one or more collaboration objects when a reproducible problem occurs. If a problem occurs only once during processing, leave the tracing function enabled continually so that the first occurrence of the failure is captured. However, leaving the tracing function enabled continually can degrade performance. Clear the trace file periodically to simplify viewing and filtering it.

Note: Use separate files for tracing and logging. Use tracing files to diagnose problems and to show the flow of an item through the Item Synchronization for Suppliers solution. Use logging files to maintain persistent records of processed data. The Log Viewer tool has trace and log file filters that enable users to view the trace or log records for a particular business object or collaboration object.

Notices and Trademarks

Proprietary Information

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

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
AS/400e
CrossWorlds
DB2
DB2 Universal Database
iSeries
Lotus
Lotus Notes
MQIntegrator
MQSeries
OS/400
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.

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

UCC and UCCnet are trademarks of Uniform Code Council, Inc., UCCnet, Inc. or both, in the United States, other countries, or both.

UCCnet Messaging is a product and/or trademark of UCCnet and is used with permission.

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

IBM WebSphere InterChange Server Version 4.2.2

IBM WebSphere Business Integration Toolset Version 4.2.2

IBM WebSphere Business Integration Adapters Version 2.4



