IBM WebSphere Business Integration Express and Express Plus for Item Synchronization

Solution Development Guide

Version 4.3.1

Note!

Before using this information and the product it supports, read the general information under "Notices and Trademarks" on page 57.

(6February2004)

This edition of this document applies to *IBM® WebSphere® Business Integration Express for Item Synchronization*Version 4.3.1, *IBM WebSphere Business Integration Express for Item Synchronization*Version 4.3.1, and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

Contents

Solution Development Guide 1
About this document 1
Audience 1
Related documents 2
Typographic conventions 2
How the Solution Development Guide is organized 2
Processing a business object: example workflows (DTD support). 6
ItemAdd workflow: adding a new item to UCCnet (DTD support). 6
ItemPublicationAdd workflow: making a new item available to trading partners and processing their responses. 8
ItemChange workflow: updating item information in UCCnet (DTD support) 13
ItemPublicationChange workflow: making updated item information available to trading partners and processing their responses 16
ItemDelist workflow: making an item permanently unavailable to trading partners (DTD support) 21
ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (DTD support) 23
Processing a business object: example workflows (schema support) 24
ItemAdd workflow: adding a new item to UCCnet (schema support) 24
CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses 27
ItemChange workflow: updating item information in UCCnet (schema support) . . . 30
CatalogueItemNotification_Change and CatalogueItemPublication_Change workflow: making updated item information available to trading partners and processing their responses . 33
ItemDelist workflow: making an item permanently unavailable to trading partners (schema support) 36
ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (schema support) 38
Checking that item data exists for fields required by UCCnet. 39
Using the PROCESSED_GTIN table 40
Using the audit_log table 41
Using the trading_partner table. 43
Polling UCCnet for worklists 43
Using subdiagrams 44
AUTHORIZATION_RESPONSES subdiagram . . 46
CATEGORY_ADD_CHANGE subdiagram . . . 46
CATALOGUE_ITEM_CONFIRMATION subdiagram 47
CIN_RESPONSE subdiagram 47
DEAD_LETTER_PUB_RECEIPT subdiagram . . 47
INITIAL_ITEM_LOAD_REQUEST subdiagram 47
ITEM_ADD_CHANGE subdiagram 48
NEW_ITEM_PUBLICATION_REQUEST subdiagram 48
PUBLICATION_COMMAND_RESPONSE subdiagram 49
RCIR_RESPONSE subdiagram 49
RCIR_QUERY_RESPONSE subdiagram 50
SIMPLE_RESPONSE subdiagram 50
UNKNOWN_MESSAGES subdiagram 51
UNKNOWN_RESPONSE subdiagram 51
Sending e-mail 51
Alerting e-mail recipients of processing errors . . 51
Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams 52
Logging 55
Tracing 55

Notices and Trademarks 57
Notices 57
Programming interface information 58
Trademarks and service marks 59

Solution Development Guide

The workflows described in this document are 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.

About this document

The products IBM® WebSphere® Business Integration Express for Item Synchronization and IBM® WebSphere® Business Integration Express Plus for Item Synchronization are made up of the following components: InterChange Server Express, the associated Toolset Express product, the Item Synchronization Collaboration, and a set of software integration adapters. Together, the components provide business process integration and connectivity among leading e-business technologies and enterprise applications as well as integration with the UCCnet GLOBAL registry.

The IBM WebSphere Business Integration Express for Item Synchronization product includes InterChange Server Express, the associated Toolset Express product, the Item Synchronization Collaboration, and a set of software integration adapters. Together they provide business process integration and connectivity among leading e-business technologies and enterprise applications as well as integration with the UCCnet GLOBALregistry.

The *Solution Development Guide* describes the internal processing of the WebSphere Business Integration Express for Item Synchronization and WebSphere Business Integration Express Plus for Item Synchronization products.

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

Audience

This document is intended for programmers who design and implement workflows using the product and who might participate in designing customizations to solutions that are based upon this product. 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 WebSphere Business Integration Express for Item Synchronization product.

Programmers must also have experience with the respective operating systems where their implementations are installed.

Related documents

The complete set of documentation available with these products describe the features and components common to all WebSphere Business Integration Express for Item Synchronization and WebSphere Business Integration Express Plus for Item Synchronization installations and includes reference material on specific components.

You can install the documentation from the following site: IBM® WebSphere® Business Integration Express for Item Synchronization InfoCenter.

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a file name, information that you type, or information that the system prints to the screen.
bold	Indicates a command name or GUI control names.
italic	Indicates an important term.
blue outline	A blue outline, which is visible only when you view a manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options where you must choose only one.
[ ]	In a syntax line, brackets surround an optional parameter.
. . .	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, option[,...] means that you can enter multiple, comma-separated options.
< >	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other.
\, /	In this document, backslashes (\) are used as the convention for directory paths. For Linux installations, substitute forward slashes (/) for backslashes. All InterChange Server Express path names are relative to the directory where the product is installed on your system.
%text% and $text	Text within percent (%) signs indicates the value of the Windows. The equivalent UNIX notation $text indicates the value of the text system variable or user variable.

How the Solution Development Guide is organized

The Solution development guide introduces the mechanics of the IBM WebSphere Business Integration Express for Item Synchronization product by first presenting sample, high-level, step-by-step workflows demonstrating how the solution handles scenarios for both DTD and Schema support.

DTD support scenarios:

ItemAdd
Described in the section "ItemAdd workflow: adding a new item to UCCnet (DTD support)" on page 6.

ItemPublicationAdd
Described in the section "ItemPublicationAdd workflow: making a new

item available to trading partners and processing their responses" on page 8. (ItemAdd and ItemPublicationAdd workflows typically occur sequentially.)

ItemChange
Described in the section "ItemChange workflow: updating item information in UCCnet (DTD support)" on page 13.

ItemPublicationChange
Described in the section "ItemPublicationChange workflow: making updated item information available to trading partners and processing their responses" on page 16. (ItemChange and ItemPublicationChange workflows typically occur sequentially.)

ItemDelist
Described in the section "ItemDelist workflow: making an item permanently unavailable to trading partners (DTD support)" on page 21.

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

Schema support scenarios:

ItemAdd
Described in the section "ItemAdd workflow: adding a new item to UCCnet (schema support)" on page 24.

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 27. (ItemAdd and CatalogueItemNotification_Add workflows typically 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 27 (ItemAdd and CatalogueItemPublication_Add workflows typically occur sequentially.)

ItemChange
Described in the section "ItemChange workflow: updating item information in UCCnet (schema support)" on page 30.

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 33. (ItemAdd and CatalogueItemNotification_Add workflows typically 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 33 (ItemAdd and CatalogueItemPublication_Add workflows typically occur sequentially.)

ItemDelist
Described in the section "ItemDelist workflow: making an item permanently unavailable to trading partners (schema support)" on page 36.

ItemWithdrawal
Described in the section “ItemWithdrawal workflow: making an item temporarily unavailable to all or selected trading partners (schema support)” on page 38.

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 39 details how a UCCnet_ItemSync collaboration object ensures that the business object to be passed to UCCnet contains data in all of the fields where UCCnet requires data.
- “Using the PROCESSED_GTIN table” on page 40 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 41 provides information about 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 43 identifies how the solution maintains data in the provided trading_partner relational table, which maintains the complete list of trading partners.
- “Polling UCCnet for worklists” on page 43 describes how the solution obtains worklists from UCCnet.
- “Using subdiagrams” on page 44 details the logic behind the subdiagrams contained in a UCCnet_processWorklist collaboration object.
- “Sending e-mail” on page 51 describes how solution collaboration objects alert e-mail recipients of processing errors, and how subdiagrams within a UCCnet_processWorklist collaboration object process mail for different processing circumstances.
- “Logging” on page 55 describes the capabilities of various collaboration objects to log errors.
- “Tracing” on page 55 outlines how problems that might occur in the solution workflow can be traced and identified.

New in this release

New in version 4.3.1: February 2004
- The text of the section“Using the PROCESSED_GTIN table” on page 40 has been revised and expanded.
- The text of the section “Using the trading_partner table” on page 43 has been revised and expanded.

December 2003

This is the second release of WebSphere Business Integration Express for Item Synchronization V4.3.1 and the first release of WebSphere Business Integration Express Plus for Item Synchronization V4.3.1.

Except where noted, all the information in this guide applies to both IBM WebSphere Business Integration Express for Item Synchronization and IBM

WebSphere Business Integration Express Plus for Item Synchronization. The term ″WebSphere Business Integration Express for Item Synchronization″ and its variants refer to both products.

Planning the configuration

Before you install and configure the IBM WebSphere Business Integration Express for Item Synchronization product, 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 will determine the connector that you use to communicate with it. If you exchange messages with UCCnet using an AS2/EDIINT interface protocol, you can use iSoftConnector or communicate with UCCnet through an iSoft Peer-to-Peer Agent. If you exchange messages through the UCCnet Command Line Utility (CLU) or are testing your installation, you can use a JTextISoftConnector.

Because the actual connector you use to communicate with UCCnet is dependent on your setup, the term “AS2 channel connector” is used throughout this document as a general term for either the iSoftConnector or JTextISoftConnector.

Note: The iSoftConnector is available only on the Windows operating system. Use the JTextISoftConnector to access the Peer-to-Peer Agent on the Linux and iSeries platforms.

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 IBM WebSphere Business Integration Express for Item Synchronization product handles the workflows that support the DTD-based implementations.

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 55 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 55 for detailed information.

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, ItemPublicationAdd detailed in the section "ItemPublicationAdd workflow: making a new item available to trading partners and processing their responses" on page 8, accomplishes sending these notifications.

Notes:
1. Mappings of some attributes in the ItemAdd messages requires the use of value translation tables. The InterChange Server Express implements these tables as cross-reference relationships.
2. If you are using schema support, refer to the documentation found in "ItemAdd workflow: adding a new item to UCCnet (schema support)" on page 24

The following instructions describe how high-level components of the Item Synchronization Collaboration 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. In this example, the SAPConnector is used. The SAPConnector converts the input from the ERP into a SAP application specific business object.
2. The SAPConnector passes the SAP application specific business object to a UCCnet_ItemSync collaboration object, first transforming it into a generic ItemBasic business object with a *Create* verb by passing it through the Sa4CwItemBasic input map.
3. The UCCnet_ItemSync collaboration object accepts the object on its *From* port and checks that required fields contain information, as detailed in the section "Checking that item data exists for fields required by UCCnet" on page 39. If all required fields are complete, the collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an mail to a configured address, as detailed in the section "Alerting e-mail recipients of processing errors" on page 51. For this example, assume all fields are complete.
4. 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 40 for more information about the PROCESSED_GTIN table.

5. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to identify the ItemAdd transaction processed. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.
6. The UCCnet_ItemSync collaboration object delivers the ItemBasic business object to the AS2 channel connector over its *To* port. In the connector controller, it is converted into an application-specific business object, a UCCnetDTD_envelope business object. The ItemBasic business object is converted by passing through the RouterMap_CwItemBasic_to_UCCnetDTD_envelope router map and CwItemBasic_to_UCCnetDTD_envelope_documentCommand_item translation map.
7. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the ItemAdd XML message in UCCnet format.
8. The AS2 channel connector passes this message to the AS2 channel server.
9. The AS2 channel server creates the digest, encrypts, and transmits the ItemAdd message to UCCnet.
10. UCCnet generates and returns to the AS2 channel server a Message Disposition Notification (MDN) to indicate successful receipt of the ItemAdd message.
11. The AS2 channel server delivers the MDN to the AS2 channel connector.
12. The AS2 channel connector sends the MDN to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object.
13. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
14. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
15. The UCCnet_processWorklist collaboration object receives the business object on its *From* port, identifies it as an MDN, and dispatches it to its SIMPLE_RESPONSE subdiagram. See the section "SIMPLE_RESPONSE subdiagram" on page 50 for more information about this subdiagram.
16. The SIMPLE_RESPONSE subdiagram sends e-mail to the recipients configured in the UCCnet_processWorklist_SIMPLE_RESPONSEObject collaboration object (an instance of the Notify_by_eMail collaboration template). See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information about how to configure properties controlling mail.
17. UCCnet creates a worklist containing the notification response for a successful ItemAdd.
18. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform) that is created during installation of the solution to the `event` directory of the JTextRWLConnector:
    - On Windows® systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
    - On Linux® and OS/400® systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

    Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

    The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section "Polling UCCnet for worklists" on page 43 for more information about this process.

19. The JTextRWLConnector retrieves the worklist query command from its `event` directory and sends it to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.
20. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope map.
21. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
22. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its *To* port. In the connector controller, it is converted to a UCCnetDTD_envelope business object. The UCCnetGBO_envelope business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_to_UCCnetDTD_envelope router and translation maps.
23. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
24. The AS2 channel connector passes the message to the AS2 channel server.
25. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.
26. UCCnet delivers the worklist containing the notification response for a successful ItemAdd to the AS2 channel server.
27. The AS2 channel server delivers the notification to the AS2 channel connector.
28. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
29. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
30. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
31. The UCCnet_processWorklist collaboration object receives the business object on its *From* port, identifies it as an ItemAdd notification, and dispatches it to its ITEM_ADD_CHANGE 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. 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 Item Synchronization Collaboration 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 6) 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. As a result, the ItemPublicationAdd workflow is described as two subflows:

- The subflow that makes the new item available to the supplier's demand-side trading partners, is described in the section "ItemPublicationAdd subflow 1: making a new item available to trading partners."
- The subflow that processes the demand-side trading partners' responses to the new item, is described in the section "ItemPublicationAdd subflow 2: processing trading partners' responses to a new item" on page 11.

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

The following instructions describe how the ItemPublicationAdd workflow makes a new item available to a supplier's demand-side trading partners. At this point in processing, the ItemAdd workflow has completed and a UCCnetGBO_envelope business object has arrived in the ITEM_ADD_CHANGE subdiagram of the UCCnet_processWorklist collaboration object.

1. The UCCnet_processWorklist collaboration object's ITEM_ADD_CHANGE configures the business object so that the router map will select the correct transformation map. See the section "ITEM_ADD_CHANGE subdiagram" on page 48 for more information about this subdiagram.
2. The AS2 channel connector maps the UCCnetGBO_envelope business object into a UCCnetDTD_envelope business object. The UCCnetGBO_envelope business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_notification_to_UCCnetDTD_envelope _publishCommand router and input maps. The business object contains the corresponding ItemPublicationAdd request. This message includes a request for UCCnet to publish the item to the trading partners listed in the message
3. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the ItemPublicationAdd XML message in UCCnet format.
4. The AS2 channel connector passes the message to the AS2 channel server.
5. The AS2 channel server creates the digest, encrypts, and transmits the ItemPublicationAdd message to UCCnet.
6. UCCnet generates and sends to the AS2 channel server an MDN indicating successful receipt of the ItemPublicationAdd message.
7. The AS2 channel server delivers the MDN to the AS2 channel connector.
8. The AS2 channel connector sends the MDN to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object.
9. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
10. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
11. The UCCnet_processWorklist collaboration object receives the business object on its *From* port, identifies it as an MDN, and dispatches it to its SIMPLE_RESPONSE subdiagram. See the section "SIMPLE_RESPONSE subdiagram" on page 50 for more information about this subdiagram.
12. The SIMPLE_RESPONSE subdiagram sends e-mail to the recipients configured in the UCCnet_processWorklist_SIMPLE_RESPONSEObject collaboration object (an instance of the Notify_by_eMail collaboration template). See the

section “Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams” on page 52 for more information about how to configure properties controlling e-mail.

13. UCCnet performs a compliance check on the data and, if all data exists in the appropriate formats, creates a worklist containing a PUB_RELEASE_NEW_ITEM notification to indicate a successful ItemPublicationAdd.
14. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform) that is created during installation of the solution to the `event` directory of the JTextRWLConnector:
    - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
    - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

    Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

    The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section “Polling UCCnet for worklists” on page 43 for more information about this process.
15. The JTextRWLConnector retrieves the worklist query command from its `event` directory and sends it to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object
16. . This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.
17. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
18. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
19. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its *To* port. In the connector controller, it is converted to a UCCnetDTD_envelope business object. The UCCnetGBO_envelope business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_to_UCCnetDTD_envelope router and translation maps.
20. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
21. The AS2 channel connector passes the message to the AS2 channel server.
22. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.
23. UCCnet delivers the worklist containing the PUB_RELEASE_NEW_ITEM notification indicating a successful ItemPublicationAdd to the AS2 channel server.
24. The AS2 channel server delivers the notification to the AS2 channel connector.
25. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.

26. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
27. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
28. The UCCnet_processWorklist collaboration object receives the object on its *From* port, identifies it as a PUB_RELEASE_NEW_ITEM, and dispatches it to its NEW_ITEM_PUBLICATION_REQUEST subdiagram. See the section "NEW_ITEM_PUBLICATION_REQUEST subdiagram" on page 48 for more information about this subdiagram.

As a result of this subflow of the ItemPublicationAdd workflow, a new item is made available to a supplier's demand-side trading partners. The ongoing subflow that alerts them of the item's existence and processes their responses, referred to as the ItemPublicationAdd subflow 2, is continued in the section "ItemPublicationAdd subflow 2: processing trading partners' responses to a new item."

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

1. The UCCnet_processWorkflow collaboration object's NEW_ITEM_PUBLICATION_REQUEST subdiagram logic:
   a. 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 40 for more information about the PROCESSED_GTIN table.
   b. Checks that the new item 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 43 for more information about the trading_partner table.
   c. Configures the business object so that the router map will be used by the connector.
   d. Sends the ItemPublicationAdd for the GTIN to the demand-side trading partners identified in the business object. The message is sent over the *NEW_ITEM_PUBLICATION_REQUEST* port to the AS2 channel connector.
   e. Logs the notification in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.
2. The AS2 channel connector maps the business object into a UCCnetDTD_envelope business object. The business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_notification_to_UCCnetDTD_envelope _publishCommand maps.
3. The business object is converted to XML, delivered to the AS2 channel server, and then to UCCnet.

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 user's existing environment and the demand-side user is ready to begin trading.

   PEND_PUBLICATION
   : The demand-side user 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 user has no interest in the product.

   PRE_AUTHORIZATION
   : The demand-side user wants to begin the process of integrating the product into its existing environment. This response might indicate that the supplier contact the demand-side user to begin the process of deciding on order quantities, pricing specifics, and so on.

   DE_AUTHORIZATION
   : The demand-side user has removed the product from the assortment and wants no further updates sent for the product.

   Note: Assume a demand-side trading partner responds with an AUTHORIZE response.

   For this example,
5. UCCnet performs a compliance check on the data. If all data exists in the appropriate formats, then UCCnet creates a worklist containing the notification response.
6. UCCnet delivers the worklist to the AS2 channel server.
7. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform) that is created during installation of the solution to the `event` directory of the JTextRWLConnector:
   - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
   - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

   Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

   The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section "Polling UCCnet for worklists" on page 43 for more information about this process.
8. The JTextRWLConnector retrieves the worklist query command from its `event` directory and sends it to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.
9. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
10. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
11. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the

AS2 channel connector over its *To* port. In the connector controller, it is converted to a UCCnetDTD_envelope business object. The UCCnetGBO_envelope business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_to_UCCnetDTD_envelope router and translation maps.

12. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
13. The AS2 channel connector passes the message to the AS2 channel server.
14. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.
15. UCCnet delivers the worklist containing the AUTHORIZE notification to the AS2 channel server.
16. The AS2 channel server delivers the notification to the AS2 channel connector.
17. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
18. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
19. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
20. The UCCnet_processWorklist collaboration object receives the object on its *From* port, identifies it as an AUTHORIZE response, and dispatches it to its AUTHORIZATION_RESPONSES subdiagram. See the section "AUTHORIZATION_RESPONSES subdiagram" on page 46 for more information about this subdiagram.
21. The AUTHORIZATION_RESPONSES subdiagram completes the following steps:
    a. Sends e-mail to the recipients configured in the UCCnet_processWorklist_AUTHORIZATION_RESPONSESObject collaboration object (an instance of the Notify_by_eMail collaboration template). See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information about how to configure properties controlling e-mail.
    b. Logs the event in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.

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, ItemPublicationChange detailed in the section "ItemPublicationChange workflow: making updated item information available to trading partners and processing their responses" on page 16, accomplishes sending these notifications.

Notes:

1. The mappings used in processing ItemChange messages use value translation tables. The InterChange Server Express implements these tables as cross-references.

2. If you are using schema support, refer to the documentation found in "ItemChange workflow: updating item information in UCCnet (schema support)" on page 30

The following instructions describe how high-level components of the Item Integration Collaboration perform the ItemChange workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector portion of an adapter specific to that ERP. In this example, the SAPConnector is used. The SAPConnector converts the input from the ERP into a SAP application specific business object.
2. The SAPConnector passes the SAP application specific business object to a UCCnet_ItemSync collaboration object, first transforming it into a generic ItemBasic business object with a *Update* verb by passing it through the Sa4CwItemBasic input map.
3. The UCCnet_ItemSync collaboration object accepts the object on its *From* port and checks that required fields contain information, as detailed in the section "Checking that item data exists for fields required by UCCnet" on page 39. If all required fields are complete, the collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an e-mail to a configured address, as detailed in the section "Alerting e-mail recipients of processing errors" on page 51. For this example, assume all fields are complete.
4. 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 completes the following steps:
     - 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 verb to *Create* and adds it 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 40 for more information about the PROCESSED_GTIN table.
5. 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 41 for more information about the audit_log table.
6. The UCCnet_ItemSync collaboration object delivers the ItemBasic business object to the AS2 channel connector over its *To* port.

   Notes:

   a. If you are using iSoftConnector, the associated map is RouterMap_CwItemBasic_to_UCCnetDTD_envelope.

   b. If you are using JTextISoftConnector, the associated map is RouterMap_CwItemBasic_to_UCCnetDTD_envelope.

In the connector controller, it is converted into a UCCnetDTD_envelope. The ItemBasic business object is converted by passing through the RouterMap_CwItemBasic_to_UCCnetDTD_envelope and CwItemBasic_to_UCCnetDTD_envelope_documentCommand_item router and translation maps.

7. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the ItemChange XML message in UCCnet format.
8. The AS2 channel connector passes the message to the AS2 channel server.
9. The AS2 channel server creates the digest, encrypts, and transmits the ItemChange message to UCCnet.
10. UCCnet generates and returns to the AS2 channel server an MDN to indicate successful receipt of the ItemChange message.
11. The AS2 channel server delivers the MDN to the AS2 channel connector.
12. The AS2 channel connector sends the MDN to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object.
13. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope.
14. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
15. The UCCnet_processWorklist collaboration object receives the business object on its *From* port, identifies it as an MDN, and dispatches it to its SIMPLE_RESPONSE subdiagram. See the section "SIMPLE_RESPONSE subdiagram" on page 50 for more information about this subdiagram.
16. The SIMPLE_RESPONSE subdiagram sends e-mail to the recipients configured in the UCCnet_processWorklist_SIMPLE_RESPONSEObject collaboration object (an instance of the Notify_by_eMail collaboration template). See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information about how to configure properties controlling e-mail.
17. UCCnet creates a worklist containing an ItemChange notification to indicate a successful ItemChange.
18. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform) that is created during installation of the solution to the `event` directory of the JTextRWLConnector:
    - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
    - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

    Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

    The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section "Polling UCCnet for worklists" on page 43 for more information about this process.
19. The JTextRWLConnector retrieves the worklist query command from its `event` directory and sends it to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.

20. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
21. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
22. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its *To* port. In the connector controller, it is converted to a UCCnetDTD_envelope business object. The UCCnetGBO_envelope business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_to_UCCnetDTD_envelope router and translation maps.
23. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
24. The AS2 channel connector passes the message to the AS2 channel server.
25. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.
26. UCCnet delivers the worklist containing the ItemChange notification for a successful ItemChange to the AS2 channel server.
27. The AS2 channel server delivers the notification to the AS2 channel connector.
28. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
29. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
30. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
31. The UCCnet_processWorklist collaboration receives the business object on its *From* port, 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. This ongoing workflow, referred to as the ItemPublicationChange workflow, is continued 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 Item Synchronization Collaboration 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 13) 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. 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 19.

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

At this point in processing, the ItemChange workflow has completed and a UCCnetGBO_envelope business object has arrived in the ITEM_ADD_CHANGE subdiagram of the UCCnet_processWorklist collaboration object.

The following instructions describe how the ItemPublicationChange workflow makes updated item information available to a supplier's demand-side trading partners:

1. The UCCnet_processWorklist collaboration object's ITEM_ADD_CHANGE subdiagram completes the following steps:
   a. The UCCnet_processWorklist collaboration object's ITEM_ADD_CHANGE subdiagram configures the business object so that the router map will select the correct transformation map.
   b. The subdiagram sends the ItemPublicationChange request over the *ITEM_ADD_CHANGE* port to the AS2 channel connector.
   c. Logs the notification in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.

   See the section "ITEM_ADD_CHANGE subdiagram" on page 48 for more information about this subdiagram.
2. The AS2 channel connector maps the UCCnetGBO_envelope business object into a UCCnetDTD_envelope business object. The UCCnetGBO_envelope business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_notification_to_UCCnetDTD_envelope_publishCommandrouter and transformation maps. The business object contains the corresponding ItemPublicationAdd request. This message includes a request for UCCnet to publish the item to the trading partners listed in the message.
3. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the ItemPublicationChange XML message in UCCnet format.
4. The AS2 channel connector passes the message to the AS2 channel server.
5. The AS2 channel server creates the digest, encrypts, and transmits the ItemPublicationChange message to UCCnet.
6. UCCnet generates and sends to the AS2 channel server an MDN indicating successful receipt of the ItemPublicationChange message.
7. The AS2 channel server delivers the MDN to the AS2 channel connector.
8. The AS2 channel connector sends the MDN to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object.
9. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.

10. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
11. The UCCnet_processWorklist collaboration object receives the business object on its *From* port, identifies it as an MDN, and dispatches it to its SIMPLE_RESPONSE subdiagram. See the section "SIMPLE_RESPONSE subdiagram" on page 50 for more information about this subdiagram.
12. The SIMPLE_RESPONSE subdiagram sends e-mail to the recipients configured in the UCCnet_processWorklist_SIMPLE_RESPONSEObject collaboration object (an instance of the Notify_by_eMail collaboration template). See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information about how to configure properties controlling e-mail.
13. UCCnet performs a compliance check on the data. If all the data are in the correct formats, UCCnet creates a worklist containing the PUB_RELEASE_DATA_CHANGE notification to indicate a successful ItemPublicationChange.
14. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform) that is created during installation of the solution to the `event` directory of the JTextRWLConnector:
    - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
    - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

    Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

    The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section "Polling UCCnet for worklists" on page 43 for more information about this process.
15. The JTextRWLConnector retrieves the worklist query command from its `event` directory and sends it to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.
16. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
17. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
18. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its *To* port. In the connector controller, it is converted to a UCCnetDTD_envelope business object. The UCCnetGBO_envelope business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_to_UCCnetDTD_envelope router and translation maps.
19. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
20. The AS2 channel connector passes the message to the AS2 channel server.
21. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.

22. UCCnet delivers the worklist containing the PUB_RELEASE_DATA_CHANGE notification for a successful ItemPublicationChange to the AS2 channel server.
23. The AS2 channel server delivers the notification to the AS2 channel connector.
24. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
25. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
26. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
27. The UCCnet_processWorklist collaboration object receives the object on its *From* port, identifies it as a PUB_RELEASE_DATA_CHANGE, and dispatches it to its NEW_ITEM_PUBLICATION_REQUEST subdiagram. See the section "NEW_ITEM_PUBLICATION_REQUEST subdiagram" on page 48 for more information about this subdiagram.

As a result of this subflow of the ItemPublicationChange workflow, updated item information is made available to a supplier's demand-side trading partners. The ongoing subflow that alerts them of the available updated information and processes their responses, referred to as the ItemPublicationChange subflow 2, is continued in the section "ItemPublicationChange subflow 2: processing trading partners' responses to updated item information."

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 instructions describe how the ItemPublicationChange workflow alerts demand-side trading partners that new item information is available and processes their responses:

1. The UCCnet_processWorkflow collaboration object's NEW_ITEM_PUBLICATION_REQUEST subdiagram logic completes the following steps:
   a. 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 40 for more information about the PROCESSED_GTIN table.
   b. Checks that the 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 43 for more information about the trading_partner table.
   c. Sends the ItemPublicationChange for the GTIN to the demand-side trading partners identified in the business object. The message is sent over the *NEW_ITEM_PUBLICATION_REQUEST* port to the AS2 channel connector.
   d. Logs the notification in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.

2. The AS2 channel connector maps the business object into a UCCnetDTD_envelope business object. The business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_notification_to_UCCnetDTD_envelope_ publishCommand router and transformation maps.
3. The business object is converted to XML, delivered to the AS2 channel server, and then to UCCnet.
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 user's existing environment and the demand-side user is ready to begin trading.

   PEND_PUBLICATION
   : The demand-side user 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 user has no interest in the product.

   PRE_AUTHORIZATION
   : The demand-side user wants to begin the process of integrating the product into its existing environment. This response might indicate that the supplier contact the demand-side user to begin the process of deciding on order quantities, pricing specifics, and so on.

   DE_AUTHORIZATION
   : The demand-side user has removed the product from the assortment and wants no further updates sent for the product.

   Note: Assume a demand-side trading partner responds with an AUTHORIZE response.
5. UCCnet performs a compliance check on the data. If all the data exist in the appropriate formats, then UCCnet creates a worklist containing the notification response.
6. UCCnet delivers the worklist to the AS2 channel server.
7. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform) that is created during installation of the solution to the `event` directory of the JTextRWLConnector:
   - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
   - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

   Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

   The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section "Polling UCCnet for worklists" on page 43 for more information about this process.
8. The JTextRWLConnector retrieves the worklist query command from its `event` directory and sends it to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.

9. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
10. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
11. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its *To* port. In the connector controller, it is converted to a UCCnetDTD_envelope business object. The UCCnetGBO_envelope business object is converted by passing through the RouterMap_UCCnetGBO_envelope_to_UCCnetDTD_envelope and UCCnetGBO_envelope_to_UCCnetDTD_envelope router and translation maps.
12. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
13. The AS2 channel connector passes the message to the AS2 channel server.
14. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.
15. UCCnet delivers the worklist containing the AUTHORIZE notification to the AS2 channel server.
16. The AS2 channel server delivers the notification to the AS2 channel connector. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetDTD_envelope business object. This business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
17. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetDTD_envelope_to_UCCnetGBO_envelope input map.
18. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
19. The UCCnet_processWorklist collaboration object receives the object on its *From* port, identifies it as an AUTHORIZE response, and dispatches it to its AUTHORIZATION_RESPONSES subdiagram. See the section "AUTHORIZATION_RESPONSES subdiagram" on page 46 for more information about this subdiagram.
20. The AUTHORIZATION_RESPONSES subdiagram completes the following steps:
    a. Sends e-mail to the recipients configured in the UCCnet_processWorklist_AUTHORIZATION_RESPONSESObject collaboration object (an instance of the Notify_by_eMail collaboration template). See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information about how to configure properties controlling e-mail.
    b. Logs the event in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.

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 23 on page 22.) 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 36

The following instructions describe how high-level components of the Item Synchronization Collaboration perform the ItemDelist workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector portion of an adapter specific to that ERP. In this example, the SAPConnector is used. The SAPConnector converts the input from the ERP into a SAP application specific business object.
2. The SAPConnector passes the SAP application specific business object to a UCCnet_ItemSync collaboration object, first transforming it into a generic ItemBasic business object with an *Delist* verb by passing it through the Sa4CwItemBasic input map.
3. The UCCnet_ItemSync collaboration object accepts the object on its *From* port and checks that required fields contain information, as detailed in the section "Checking that item data exists for fields required by UCCnet" on page 39. If all required fields are complete, the collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an e-mail to a configured address, as detailed in the section "Alerting e-mail recipients of processing errors" on page 51. For this example, assume all fields are complete.
4. The UCCnet_ItemSync collaboration object removes the item from the PROCESSED_GTIN table. See the section "Using the PROCESSED_GTIN table" on page 40 for more information about the PROCESSED_GTIN table.
5. 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 41 for more information about the audit_log table.
6. The UCCnet_ItemSync collaboration object delivers the ItemBasic business object to the AS2 channel connector over its *To* port. In the connector controller, it is converted to a UCCnetDTD_envelope business object. The ItemBasic business object is converted by passing through the RouterMap_CwItemBasic_to_UCCnetDTD_envelope and CwItemBasic_to_UCCnetDTD_envelope_publishCommand_documentIdentifier router and translation maps.
7. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the ItemDelist XML message in UCCnet format.
8. The AS2 channel connector passes this message to the AS2 channel server.
9. The AS2 channel server creates the digest, encrypts, and transmits the ItemDelist message to UCCnet.
10. UCCnet generates an MDN to indicate successful receipt of the ItemDelist message.

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. 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 (DTD support)" on page 21.) 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 38

The following instructions describe how high-level components of the Item Synchronization Collaboration perform the ItemWithdrawal workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector portion of an adapter specific to that ERP. In this example, the SAPConnector is used. The SAPConnector converts the input from the ERP into a SAP application specific business object.
2. The SAPConnector passes the SAP application specific business object to a UCCnet_ItemSync collaboration object, first transforming it into a generic ItemBasic business object with an *Withdraw* verb by passing it through the Sa4CwItemBasic input map.
3. The UCCnet_ItemSync collaboration object accepts the object on its *From* port and checks that required fields contain information, as detailed in the section "Checking that item data exists for fields required by UCCnet" on page 39. If all required fields are complete, the collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an e-mail to a configured address, as detailed in the section "Alerting e-mail recipients of processing errors" on page 51. For this example, assume all fields are complete.
4. The UCCnet_ItemSync collaboration object locates the item in the PROCESSED_GTIN table and sets the value for the **withdrawn** field to Y. This action prevents the publication of the item in response to an incoming INITIAL_ITEM_LOAD_REQUEST. See the section "Using the PROCESSED_GTIN table" on page 40 for more information about the PROCESSED_GTIN table.
5. 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 41 for more information about the audit_log table.
6. The UCCnet_ItemSync collaboration object delivers the ItemBasic business object to the AS2 channel connector over its *To* port. In the connector controller, it is converted into a UCCnetDTD_envelope business object. The ItemBasic business object is converted by passing through the RouterMap_CwItemBasic_to_UCCnetDTD_envelope and CwItemBasic_to_UCCnetDTD_envelope_publishCommand_documentIdentifier router and translation maps.

7. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the ItemWithdrawal XML message in UCCnet format.
8. The AS2 channel connector passes this message to the AS2 channel server.
9. The AS2 channel server creates the digest, encrypts, and transmits the ItemWithdrawal message to UCCnet.
10. UCCnet generates and returns to the AS2 channel server an MDN to indicate successful receipt of the ItemWithdrawal message.

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 (schema support)

The information in the following sections outlines at a high level how the IBM WebSphere Business Integration Express for Item Synchronization product handles the workflows that support the schema-based implementations.

Collaboration objects based on the ItemCommandRouter, Notify_by_eMail, UCCnet_ItemSync, UCCnet_requestWorklist, UCCnet_processWorklist, or CIN_CIP_Dispatcher collaboration templates log error messages if they encounter error situations during any stage of processing. See the section "Logging" on page 55 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 55 for detailed information.

Notes:
1. If you are using iSoftConnector, the associated map for the first instance is CwItemBasic_to_UCCnetXSD_envelope_registerCommand_itemAddChange. If you are setting up for XSD CIP operation, you need two instances of iSoftConnector. The associated map for the second instance is CwItemBasic_to_UCCnetXSD_envelope_publicationCommand_catalogueItem.
2. If you are using JTextISoftConnector, the associated map for the first instance is CwItemBasic_to_UCCnetXSD_envelope_to_registerCommand_itemAddChange. If you are setting up for XSD CIP operation, you need two instances of JTextISoftConnector. The associated map for the second instance is CwItemBasic_to_UCCnetXSD_envelope_publicationCommand_catalogueItem.

ItemAdd workflow: adding a new item to UCCnet (schema 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 27, send these notifications.

Notes:
1. Mappings of some attributes in the ItemAdd messages requires the use of value translation tables. The InterChange Server Express implements these tables as cross-reference relationships.
2. If you are not using schema support, refer to the documentation found in "ItemAdd workflow: adding a new item to UCCnet (DTD support)" on page 6.

The following instructions describe how high-level components of the Item Synchronization Collaboration perform the ItemAdd workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector portion of an adapter specific to that ERP. In this example, the SAPConnector is used. The SAPConnector converts the input from the ERP into a SAP application specific business object.
2. The SAPConnector passes the SAP application specific business object to a UCCnet_ItemSync collaboration object, first transforming it into a generic ItemBasic business object with a *Create* verb by passing it through the Sa4CwItemBasic input map.
3. The UCCnet_ItemSync collaboration object accepts the object on its *From* port and checks that required fields contain information, as detailed in the section "Checking that item data exists for fields required by UCCnet" on page 39. If all required fields are complete, the collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an e-mail to a configured address, as detailed in the section "Alerting e-mail recipients of processing errors" on page 51. For this example, assume all fields are complete.
4. 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 40 for more information about the PROCESSED_GTIN table.
5. The UCCnet_ItemSync collaboration object adds an entry to the audit_log table to identify the ItemAdd transaction processed. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.
6. The UCCnet_ItemSync collaboration object delivers the ItemBasic business object to the ItemCommandRouter collaboration object.
7. The ItemCommandRouter collaboration object receives the business object on its *From* port and determines that it represents an item to be added to the UCCnet. It them passes the business object to its *ToRCIR* port.
8. The ItemBasic business object is passed to the AS2 channel connector, where it is converted into an application specific business object in the connector controller by the CwItemBasic_to_UCCnetXSD_envelope_registerCommand_itemAddChange map.
9. After the conversion is complete, the business object is passed to the AS2 channel connector agent, which sends it to the Data Handler for XML to produce the ItemAdd XML message in UCCnet format.
10. The connector then passes the message to the AS2 channel server.
11. The server created the digest, encrypts, and then transmits the ItemAdd message to UCCnet.
12. UCCnet creates a worklist containing the notification response RCIR_ADD_Response.
13. A time-triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform), which is created during installation of the solution, and send them to the `event` directory of the JTextRWLConnector:
    - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
    - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section "Polling UCCnet for worklists" on page 43 for more information about this process.

14. The JTextRWLConnector retrieves the worklist query command from its `event` directory and sends it to the Data Handler for XML, which converts it into a UCCnetXSD_envelope business object. This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.
15. The business object is converted to a UCCnetGBO_envelope by the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
16. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
17. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its *To* port. In the connector controller, it is converted into a UCCnetXSD_envelope. The UCCnetGBO_envelope business object is converted by passing through the UCCnetGBO_envelope_to_UCCnetXSD_envelope input map.
18. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
19. The AS2 channel connector passes the message to the AS2 channel server.
20. The AS2 channel server creates the digest, and encrypts and transmits the message to UCCnet.
21. UCCnet delivers the worklist containing the RCIR_RESPONSE notification indicating a successful ItemAdd to the AS2 channel server.
22. The AS2 channel server delivers the notification to the AS2 channel connector.
23. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetXSD_envelope business object. The business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
24. The business object is converted to a UCCnetGBO_envelope business object by the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
25. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
26. 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 of the UCCnet_processWorklist collaboration object.

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 workflows that accomplish this are described in the section "CatalogueItemNotification_Add and CatalogueItemPublication_Add workflows: making a new item available to trading partners and processing their responses" on page 27.

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 Item Synchronization Collaboration 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 (schema support)" on page 24) 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. At this point in processing, 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 RCIR_RESPONSE notification inside a UCCnetGBO_envelope business object for the received message. It logs the notification in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.
2. It creates an empty ItemBasic business object and sends it to the DestinationAppRetrieve port with a retrieve verb.
3. The ItemBasic business object that initiated the corresponding RCIR command, which was sent it to UCCnet, is returned on the DestinationAppRetrieve port with a Create verb.
4. The ItemBasic BO is sent to the RCIR_RESPONSE port with a Create verb.
5. When using a supplier-implemented data source pool:
   a. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port and maps it to a CatalogueItemNotification_ADD UCCnetGBO_envelope using the CwItemBasic_to_UCCnetGBO_envelope_notifyCommand _catalogueItem map.
   b. 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.
   c. The collaboration object sends a CatalogueItemNotification_ADD (CIN_ADD) notification out the To port to the AS2 channel connector for each GLN found in the GLN subscription file.
   d. The UCCnetGBO_envelope business object is mapped to a UCCnetXSD_envelope.
   e. The AS2 channel connector sends the business objects to the Data Handler for XML, which produces the CIN_ADD XML messages in UCCnet format.
   f. The AS2 channel connector passes the messages to the AS2 channel server.
   g. The AS2 channel server creates the digest, encrypts and transmits the messages to UCCnet.
   h. UCCnet forwards the CIN_ADD messages to the demand-side trading partners.

   When using UCCnet as the data source pool:
   a. The AS2 channel connector receives the ItemBasic business object and maps it to a CatalogueItem_ADD UCCnetXSD_envelope business object using CwItemBasic_to_UCCnetXSD_envelope_publicationCommand _catalogueItem.
   b. The AS2 channel connector sends the business objects to the Data Handler for XML, which produces the CI_ADD XML message in UCCnet format.

c. The AS2 channel connector passes the message to the AS2 channel server.
d. The AS2 channel server creates the digest, encrypts and transmits the message to UCCnet.
e. UCCnet returns a catalogue item response to the AS2 channel server.
f. The AS2 channel server delivers the notification to the AS2 channel connector.
g. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetXSD_envelope. The business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
h. The business object is converted to a UCCnetGBO_envelope business object by the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
i. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
j. The UCCnet_processWorklist collaboration object receives the business object on its From port, identifies it as an CI response notification, and sends it to the PUBLICATION_COMMAND_RESPONSE subdiagram of the UCCnet_processWorklist collaboration object.
k. The PUBLICATION_COMMAND_RESPONSE subdiagram creates an empty ItemBasic business object and sends it to the DestinationAppRetrieve port with a retrieve verb
l. The ItemBasic business object that initiated the corresponding RCIR command, which was sent to UCCnet, is returned.
m. The returned ItemBasic business object is sent to the PUBLICATION_CMD_RESPONSE port with a Create verb.
n. 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 using the CwItemBasic_to_UCCnetGBO_env_publicationCommand _catalogueItemPublication map.
o. It uses the business object's category code to retrieve the GLNs of any subscribed trading partners from the GLN subscription file. This file is determined by the DISPATCHER_GLN_FILE property.
p. The collaboration object sends a CatalogueItemPublication_ADD (CIP_ADD) notification out the To port to the AS2 channel connector for each GLN found in the GLN subscription file.
q. The UCCnetGBO_envelope is mapped to a UCCnetXSD_envelope.
r. The AS2 channel connector sends the business objects to the Data Handler for XML, which produces the CIP_ADD XML messages in UCCnet format.
s. The AS2 channel connector passes the messages to the AS2 channel server.
t. The AS2 channel server creates the digest, encrypts and transmits the messages to UCCnet.
u. UCCnet returns a catalogue item publication response to the AS2 channel server.
v. The AS2 channel server delivers the notification to the AS2 channel connector.
w. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetXSD_envelope. The business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.

x. The business object is converted to a UCCnetGBO_envelope business object by the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.

y. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.

z. The UCCnet_processWorklist collaboration object receives the business object on its From port, identifies it as a CIP response notification, and sends it to the PUBLICATION_COMMAND_RESPONSE subdiagram of the UCCnet_processWorklist collaboration object.

aa. The PUBLICATION_COMMAND_RESPONSE subdiagram sends to UCCnetGBO_envelope CIP response to the CIP_RESPONSE port with a create verb to send a notification e-mail to the supplier.

ab. As a result of receiving the CIP_ADD messages, UCCnet sends out a CatalogueItemNotification to each subscribing demand-side trading partner for whom a CIP was received.

6. The trading partners can respond with any of the following Catalogue Item Confirmation responses:

   REVIEW
   This state is used to tell parties that an item is being reviewed by the retailer.

   REJECTED
   This state is used to tell parties that an item is rejected and that no additional information is requested at this time.

   ACCEPTED
   This state is used to tell initiators that an item has been accepted by the retailer, but has not yet been synchronized. This state is similar to a DTD-based UCCnet PRE-AUTHORIZATION.

   SYNCHRONISED
   This state is used to tell initiators that an item has been accepted by the retailer and will be synchronized. This state is similar to a DTD-based UCCnet AUTHORIZE.

   Note: Assume a demand-side trading partner responds with a SYNCHRONISED response.

7. 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 a SYNCHRONISED notification to indicate a successful receipt.

8. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform), which is created during installation of the solution, to the `event` directory of the JTextRWLConnector:
   - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
   - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

   Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

   The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section "Polling UCCnet for worklists" on page 43 for more information about this process.

9. The JTextRWLConnector receives the worklist query command from its `event` directory and sends it to the Data Handler for XML.
10. The Data Handler for XML converts the worklist query command into a UCCnetXSD_envelope business object. This business object contains the entire UCCnet message, including each individual data instance and the command related to it.
11. The business object is converted to a UCCnetGBO_envelope business object by passing it through the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
12. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
13. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its To port. In the connector controller, it is converted to a UCCnetXSD_envelope business object.
14. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
15. The AS2 channel connector passes the message to the AS2 channel server.
16. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.
17. UCCnet delivers the worklist containing the SYNCHRONISED notification to the AS2 channel server.
18. The AS2 channel server delivers the notification to the AS2 channel connector.
19. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnet_XSD_envelope business object. This business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
20. The business object is converted to a UCCnetGBO_envelope business object by passing it through the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
21. The AS2 channel connector delivers the business object to the UCCnet_processWorklist collaboration object.
22. The UCCnet_processWorklist collaboration object receives the object on its From port, identifies it as a CATALOGUE_ITEM_CONFIRMATION response, and dispatches it to its CATALOGUE_ITEM_CONFIRMATION subdiagram.
23. The CATALOGUE_ITEM_CONFIRMATION subdiagram carries out the following tasks:
    - It sends an e-mail to the recipients defined by the UCCnet_processWorklist_CATALOGUE_ITEM_CONFIRMATIONObject business object, which is an instance of the Notify_by_eMail collaboration template. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information about how to configure properties controlling e-mail.
    - It logs the event in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit log table.

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 33, accomplish sending these notifications.

The following instructions describe how high-level components of the Item Synchronization Collaboration perform the ItemChange workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector portion of an adapter specific to that ERP. In this example, the SAPConnector is used. The SAPConnector converts the input from the ERP into a SAP application specific business object.
2. The SAPConnector passes the SAP application specific business object to a UCCnet_ItemSync collaboration object, first transforming it into a generic ItemBasic business object with an *Update* verb by passing it through the Sa4CwItemBasic input map.
3. The UCCnet_ItemSync collaboration object accepts the object on its *From* port and checks that required fields contain information, as detailed in the section "Checking that item data exists for fields required by UCCnet" on page 39. If all required fields are complete, the collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an e-mail to a configured address, as detailed in the section "Alerting e-mail recipients of processing errors" on page 51. For this example, assume all fields are complete.
4. 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 completes the following steps:
     – 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 verb to *Create* and adds it to the PROCESSED_GTIN table, setting the entry's **withdrawn** field to `N`.

   Assume for this example that the item already exists in the table and is not withdrawn. See the section "Using the PROCESSED_GTIN table" on page 40 for more information about the PROCESSED_GTIN table.
5. 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 41 for more information about the audit_log table.
6. The UCCnet_ItemSync collaboration object sends the ItemBasic business object through its To port to the ItemCommandRouter collaboration object.
7. The ItemCommandRouter collaboration object receives the business object on its *From* port, determines that it represents an ItemChange and sends it to its ToRCIR port.
8. The ItemBasic business object is sent to the AS2 channel connector.
9. The connector controller converts it into an application specific business object by using the

CwItemBasic_to_UCCnetXSD_envelope_registerCommand_itemAddChange map. It then sends the business object to the Data Handler for XML, which produces the ItemChange XML message in UCCnet format.

10. The AS2 channel connector passes this message to the AS2 channel server.
11. The AS2 channel server creates the digest, encrypts, and transmits the ItemChange message to UCCnet.
12. UCCnet creates a worklist containing an RCIR_CHANGE_Response notification to indicate a successful ItemChange.
13. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependant on platform), which is created during installation of the solution, to the `event` directory of the JTextRWLConnector:
    - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
    - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

    Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.

    The JTextRWLConnector polls its `event` directory for any worklist query commands that might have been delivered. See the section "Polling UCCnet for worklists" on page 43 for more information about this process.
14. The JTextRWLConnector retrieves the worklist query command from its `event` directory and sends it to the Data Handler for XML, which converts it into a UCCnetXSD_envelope application specific business object. This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.
15. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
16. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
17. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its To port. In the connector controller it is converted to a UCCnetXSD_envelope by passing through the UCCnetGBO_envelope_to_UCCnetXSD_envelope input map.
18. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
19. The AS2 channel connector passes the message to the AS2 channel server.
20. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.
21. UCCnet delivers the worklist, which contains the RCIR_CHANGE_Response notification for a successful ItemChange, to the AS2 channel server.
22. The AS2 channel server delivers the notification to the AS2 channel connector.
23. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetXSD_envelope application-specific business object.
24. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.

25. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
26. The UCCnet_processWorklist collaboration object receives the business object on its From port, identifies it as an 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. This is accomplished by the CatalogueItemNotification_Change and CatalogueItemPublication_Change workflows, 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 Item Synchronization Collaboration perform the CatalogueItemNotification_Change and CatalogueItemPublication_Change workflows. In the CatalogueItemNotification_Change and CatalogueItemPublication_Change 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 30) 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.

At this point in processing, 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 RCIR_RESPONSE notification inside a UCCnetGBO_envelope business object for the received message. It logs the notification in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit_log table.
2. It creates an empty ItemBasic business object and sends it to the DestinationAppRetrieve port with a retrieve verb.
3. The ItemBasic business object that initiated the corresponding RCIR command, which was sent it to UCCnet, is returned on the DestinationAppRetrieve port with an Update verb.
4. The ItemBasic BO is sent to the RCIR_RESPONSE port with an Update verb.
5. When using a supplier-implemented data source pool:
   a. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port and maps it to a CatalogueItemNotification_ADD UCCnetGBO_envelope using the CwItemBasic_to_UCCnetGBO_envelope_notifyCommand _catalogueItem map.
   b. 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.

c. The collaboration object sends a CatalogueItemNotification_CHANGE (CIN_CHANGE) notification out the To port to the AS2 channel connector for each GLN found in the GLN subscription file.
d. The UCCnetGBO_envelope business object is mapped to a UCCnetXSD_envelope.
e. The AS2 channel connector sends the business objects to the Data Handler for XML, which produces the CIN_CHANGE XML messages in UCCnet format.
f. The AS2 channel connector passes the messages to the AS2 channel server.
g. The AS2 channel server creates the digest, encrypts and transmits the messages to UCCnet.
h. UCCnet forwards the CIN_CHANGE messages to the demand-side trading partners.

When using UCCnet as the data source pool:

a. The AS2 channel connector receives the ItemBasic business object and maps it to a CatalogueItem_CHANGE UCCnetXSD_envelope business object using .
b. The AS2 channel connector sends the business objects to the Data Handler for XML, which produces the CI_ADD XML message in UCCnet format.
c. The AS2 channel connector passes the message to the AS2 channel server.
d. The AS2 channel server creates the digest, encrypts and transmits the message to UCCnet.
e. UCCnet returns a catalogue item response to the AS2 channel server.
f. The AS2 channel server delivers the notification to the AS2 channel connector.
g. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnetXSD_envelope. The business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
h. The business object is converted to a UCCnetGBO_envelope business object by the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
i. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.
j. The UCCnet_processWorklist collaboration object receives the business object on its From port, identifies it as an CI response notification, and sends it to the PUBLICATION_COMMAND_RESPONSE subdiagram of the UCCnet_processWorklist collaboration object.
k. The PUBLICATION_COMMAND_RESPONSE subdiagram creates an empty ItemBasic business object and sends it to the DestinationAppRetrieve port with a retrieve verb
l. The ItemBasic business object that initiated the corresponding RCIR command, which was sent to UCCnet, is returned.
m. Because this is an item chance, no CatalogueItemPublications are sent. The PUBLICATION_COMMAND_RESPONSE subdiagram sends to UCCnetGBO_envelope CI response to the CI_RESPONSE port with a create verb to send a notification e-mail to the supplier.
n. The list of suppliers that receive a notification is determined based on the CIP messages previously 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.

6. The trading partners can respond with any of the following Catalogue Item Confirmation responses:

   REVIEW
   This state is used to tell parties that an item is being reviewed by the retailer.

   REJECTED
   This state is used to tell parties that an item is rejected and that no additional information is requested at this time.

   ACCEPTED
   This state is used to tell initiators that an item has been accepted by the retailer, but has not yet been synchronized. This state is similar to a DTD-based UCCnet PRE-AUTHORIZATION.

   SYNCHRONISED
   This state is used to tell initiators that an item has been accepted by the retailer and will be synchronized. This is similar to the UCCnet Authorization.

   Note: Assume a demand-side trading partner responds with a SYNCHRONISED response.
7. 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 the SYNCHRONISED notification response.
8. A chronologically triggered process must be configured to move query command messages tailored to retrieve specific UCCnet notifications from the following directory (dependent on platform), which is created during installation of the solution, to the `event` directory of the JTextRWLConnector:
   - On Windows systems: `..\WebSphereICS\UCCnet\UCCnetMessages\Source`
   - On Linux and OS/400 systems: `../WebSphereICS/UCCnet/UCCnetMessages/Source`

   Note: This process is not part of the solution and must be customized by the user. The installation path is dependent on the path set when the solution is installed.
9. The JTextRWLConnector polls its `event` directory for any worklist query commands that have been delivered. See the section “Polling UCCnet for worklists” on page 43 for more information about this process.
10. The JTextRWLConnector receives the worklist query command from its `event` directory and sends it to the Data Handler for XML.
11. The Data Handler for XML converts the worklist query command into a UCCnetXSD_envelope business object. This business object contains the entire UCCnet message, including each individual data instance and the command related to it.
12. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
13. The JTextRWLConnector passes the business object to a UCCnet_requestWorklist collaboration object.
14. The UCCnet_requestWorklist collaboration object receives the UCCnetGBO_envelope business object on its *From* port and passes it to the AS2 channel connector over its To port. In the connector controller, it is converted to a UCCnetXSD_envelope business object.

15. The AS2 channel connector sends the business object to the Data Handler for XML, which produces the XML message in UCCnet format.
16. The AS2 channel connector passes the message to the AS2 channel server.
17. The AS2 channel server creates the digest, encrypts, and transmits the message to UCCnet.
18. UCCnet delivers the worklist containing the SYNCHRONISED notification to the AS2 channel server.
19. The AS2 channel server delivers the notification to the AS2 channel connector.
20. The AS2 channel connector sends the notification to the Data Handler for XML, which converts it into a UCCnet_XSD_envelope business object. This business object contains the entire UCCnet notification, including each individual data instance and the commands related to it.
21. The business object is converted to a UCCnetGBO_envelope business object by passing through the UCCnetXSD_envelope_to_UCCnetGBO_envelope input map.
22. The AS2 channel connector delivers the business object to the UCCnet_processWorklist collaboration object.
23. The UCCnet_processWorklist collaboration object receives the object on its From port, identifies it as a CATALOGUE_ITEM_CONFIRMATION response, and dispatches it to its CATALOGUE_ITEM_CONFIRMATION subdiagram.
24. The CATALOGUE_ITEM_CONFIRMATION subdiagram carries out the following tasks:
    - It sends an e-mail to the recipients defined by the UCCnet_processWorklist_CATALOGUE_ITEM_CONFIRMATIONObject business object, which is an instance of the Notify_by_eMail collaboration template. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information about how to configure properties controlling e-mail.
    - It logs the event in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about the audit log table.

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 38.) 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 21

The following instructions describe how high-level components of the Item Synchronization Collaboration perform the ItemDelist workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector portion of an adapter specific to that ERP. In this example, the SAPConnector is used. The SAPConnector converts the input from the ERP into a SAP application specific business object.

2. The SAPConnector passes the SAP application specific business object to a UCCnet_ItemSync collaboration object, first transforming it into a generic ItemBasic business object with an *Delist* verb by passing it through the Sa4CwItemBasic input map.
3. The UCCnet_ItemSync collaboration object accepts the object on its *From* port and checks that required fields contain information, as detailed in the section "Checking that item data exists for fields required by UCCnet" on page 39. If all required fields are complete, the collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an e-mail to a configured address, as detailed in the section "Alerting e-mail recipients of processing errors" on page 51. For this example, assume all fields are complete.
4. The UCCnet_ItemSync collaboration object removes the item from the PROCESSED_GTIN table. See the section "Using the PROCESSED_GTIN table" on page 40 for more information about the PROCESSED_GTIN table.
5. 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 41 for more information about the audit_log table.
6. The UCCnet_ItemSync collaboration object sends the ItemBasic business object through its *To* port to the ItemCommandRouter collaboration object.
7. The ItemCommandRouter receives the business object on its From port, and determines that it is a delist.
8. The ItemCommandRouter sends the ItemBasic business object through its ToCIN_CI port.
9. When using a supplier-implemented data source pool: A Notify_by_eMail collaboration object receives the ItemBasic business object and sends a note to the supplier saying that the CI_DELIST is unsupported.

   When using UCCnet as the data source pool:
   a. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port.
   b. The collaboration object maps the ItemBasic business object to a UCCnetXSD_envelope, using the map defined by its collaboration parameters.
   c. The collaboration object uses the category code from the new UCCnetXSD_envelope to retrieve the GLNs of any trading partners that subscribe to it. The collaboration object retrieves the trading partner subscription list from the GLN subscription file, defined by the CIN_DISPATCHER_GLN_FILE property.
   d. The collaboration object sends a CatalogueItemNotification_DELIST (CIN_DELIST) business object to the AS2 connector for each GLN found in the GLN subscription file.
   e. At the connector, they are converted into UCCnetXSD_envelope business objects.
   f. The AS2 channel connector sends each business object to the Data Handler for XML, which produces the CatalogueItemNotification_delist XML message in UCCnet format.
   g. The AS2 channel connector passes this message to the AS2 channel server.
   h. The AS2 channel server creates the digest, encrypts, and transmits the CatalogueItemNotification_delist message to UCCnet.
   i. UCCnet generates an MDN to indicate successful receipt of the CatalogueItemNotification_delist message.

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 36.) 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 23

The following instructions describe how high-level components of the Item Synchronization Collaboration perform the ItemWithdrawal workflow:

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the connector portion of an adapter specific to that ERP. In this example, the SAPConnector is used. The SAPConnector converts the input from the ERP into a SAP application specific business object.
2. The SAPConnector passes the SAP application specific business object to a UCCnet_ItemSync collaboration object, first transforming it into a generic ItemBasic business object with a Withdraw verb by passing it through the Sa4CwItemBasic input map.
3. The UCCnet_ItemSync collaboration object accepts the object on its From port and checks that required fields contain information, as detailed in the section "Checking that item data exists for fields required by UCCnet" on page 39. If all required fields are complete, the collaboration object continues processing it. If all required fields are not complete, the collaboration object aborts processing and sends an e-mail to a configured address, as detailed in the section "Alerting e-mail recipients of processing errors" on page 51. For this example, assume all fields are complete.
4. 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 40 for more information about the PROCESSED_GTIN table.
5. 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 41 for more information about the audit_log table.
6. The UCCnet_ItemSync collaboration object delivers the ItemBasic business object to the *From* port of the ItemCommandRouter collaboration object by sending it out from its *To* port.
7. The ItemCommandRouter collaboration object uses a combination of the verb and the DeleteFlag attribute of the ItemBasic business object to determine that the item is being withdrawn.

8. The ItemCommandRouter sends the ItemBasic business object out through its ToCIN_CI port
9. When using a supplier-implemented data source pool:
   a. The CIN_CIP_Dispatcher collaboration object receives the ItemBasic business object on its From port and maps it to a CatalogueItemNotification_WITHDRAW UCCnetGBO_envelope using the CwItemBasic_to_UCCnetGBO_envelope_notifyCommand _catalogueItem map.
   b. 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.
   c. The collaboration object sends a CatalogueItemNotification_WITHDRAW (CIN_WITHDRAW) notification out the To port to the AS2 channel connector for each GLN found in the GLN subscription file.
   d. The UCCnetGBO_envelope business object is mapped to a UCCnetXSD_envelope.
   e. The AS2 channel connector sends the business objects to the Data Handler for XML, which produces the CIN_WITHDRAW XML messages in UCCnet format.
   f. The AS2 channel connector passes the messages to the AS2 channel server.
   g. The AS2 channel server creates the digest, encrypts and transmits the messages to UCCnet.
   h. UCCnet forwards the CIN_WITHDRAW messages to the demand-side trading partners.

   When using UCCnet as the data source pool:A Notify_by_eMail collaboration object receives the ItemBasic business object and sends a note to the supplier saying the CI_DELIST 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.

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 (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 e-mail requesting the missing information to an e-mail address provided in the UCCnet_ItemSync collaboration object's SEND_MAIL_TO configuration property. See the section "Alerting e-mail recipients of processing errors" on page 51 for more information about how e-mail is handled within the solution.

Using the PROCESSED_GTIN table

The PROCESSED_GTIN table is a relational table provided with the IBM WebSphere Business Integration Express for Item Synchronization product. 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 47 for more information about 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 that UCCnet requires data for, are complete (see the section "Checking that item data exists for fields required by UCCnet" on page 39 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 completes the following steps:
    - 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 the Quick Start Guide for instructions on how to create these collaboration objects and set each object's collaboration-specific 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 is created by running the supplied InitializeRelationshipTables.sql file for the database type (DB2® or Microsoft® SQL Server). See the Quick Start Guide for information about running the InitialRelationshipTables.sql file to populate the relationships after deployment.

Using the audit_log table

The audit_log table is provided with the IBM WebSphere Business Integration Express for Item Synchronization product. 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 what trading partners are participating in the trading community and with what products that participation is associated.

The audit service is composed of three components:
- The audit_log table receives log entries from each participant component in the solution. Some possible fields and their corresponding values are shown in the following table.

Table 1. Log entry samples

Fields	Values
LOG_SEQ_NO	1
LOG_SOURCE_NAME	UCCnet2
GLN_CODE	NA

Table 1. Log entry samples (continued)

Fields	Values
SOURCE_SYSTEM	My Company Name
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 collaborations and adapters to log critical events to the audit_log table.
- 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 about how these subdiagrams operate:
- “AUTHORIZATION_RESPONSES subdiagram” on page 46
- “CATALOGUE_ITEM_CONFIRMATION subdiagram” on page 47
- “INITIAL_ITEM_LOAD_REQUEST subdiagram” on page 47
- “ITEM_ADD_CHANGE subdiagram” on page 48
- “NEW_ITEM_PUBLICATION_REQUEST subdiagram” on page 48
- “RCIR_RESPONSE subdiagram” on page 49

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 or Microsoft SQL Server. Table creation is performed via running the supplied audit_log.sql file for the database type (DB2 or Microsoft SQL Server).

Using the trading_partner table

The trading_partner table, or GLN table, is a relational table provided with the IBM WebSphere Business Integration Express for Item Synchronization product. 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 trading_partner table 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, which includes adding the following columns:

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

Note: After the columns are created, you must manually populate it with the correct information. See the Quick Start Guide for information about populating the trading_partner table.

Polling UCCnet for worklists

UCCnet never spontaneously sends notification messages; instead, UCCnet must be polled for these messages via query command messages tailored to retrieve the specific UCCnet notifications.

A chronologically triggered process must be configured to move your query command messages to the `event` directory of the JTextRWLConnector. This process is not part of the solution and must be customized by the user. The JTextRWLConnector polls its event directory for any worklist query commands that might have been delivered at a polling interval, which is set by the user. When the JTextRWLConnector finds a worklist query command, it sends the command to the Data Handler for XML, which converts the command into a UCCnet*XXX*_envelope business object.

Note: In this and the following example names, the variable *XXX* specifies the XML definition type used (DTD or XSD).

This business object contains the entire UCCnet message, including each individual data instance and the commands related to it.

The JTextRWLConnector then passes this business object to a UCCnet_requestWorklist collaboration object, which passes it to the AS2 channel server for transmission to UCCnet. UCCnet responds with the appropriate worklist, which initiates ongoing workflows in the solution.

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).

Using subdiagrams

All messages initiated from UCCnet are in UCCnet XML format. Because 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 to it a UCCnetGBO_envelope business object. This business object is created from the following process:

1. The AS2 channel connector receives the UCCnet worklist document (envelope) from the AS2 channel server.
2. The AS2 channel connector sends the document to the Data Handler for XML, which converts it into a UCCnet*XXX*_envelope business object. In these business object names, *XXX* is either `DTD` or `XSD`, depending on whether you are using a DTD XML definition or a schema-based XML definition.
3. The business object is converted to a UCCnetGBO_envelope business object by passing through an input map of the form UCCnet*XXX*_envelope_to_UCCnetGBO_envelope.
4. The AS2 channel connector delivers the business object to a UCCnet_processWorklist collaboration object.

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 46 for more information about this subdiagram.

CATALOGUE_ITEM_CONFIRMATION
Handles the process of the CATALOGUE_ITEM_CONFIRMATION responses, received by the UCCnet_processWorklist collaboration object. See the section "CATALOGUE_ITEM_CONFIRMATION subdiagram" on page 47 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 46 for more information about this subdiagram.

CIN_RESPONSE subdiagram
Handles incoming messages that are recognized as CIN_RESPONSE messages. See the section "CIN_RESPONSE subdiagram" on page 47 for more information about 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 47 for more information about 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 47 for more information about 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 48 for more information about 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 48 for more information about 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 49 for more information about this subdiagram.

RCIR_RESPONSE
Handles notifications for the ADD and CHANGE topics. See the section "RCIR_RESPONSE subdiagram" on page 49 for more information.

RCIR_QUERY_RESPONSE
Handles incoming messages that are recognized as RCIR_QUERY_RESPONSE messages. See the section "RCIR_QUERY_RESPONSE subdiagram" on page 50 for more information.

SIMPLE_RESPONSE subdiagram
Handles immediate responses to commands such as MDNs from UCCnet. See the section "SIMPLE_RESPONSE subdiagram" on page 50 for more information about this subdiagram.

UNKNOWN_MESSAGES subdiagram
Handles incoming messages not recognized as supported. See the section "UNKNOWN_MESSAGES subdiagram" on page 51 for more information about 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 51 for more information about 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 completes the following steps:

1. Instantiates a collaboration object based on the Notify_by_eMail collaboration template called UCCnet_processWorklist_AUTHORIZATION_RESPONSESObject, which sends an e-mail to one or more selected recipients. The e-mail message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information.
2. Logs the event in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about 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 e-mail containing the category maintenance information to selected recipients by instantiating a collaboration object based on the Notify_by_eMail collaboration template called UCCnet_processWorklist_CATEGORY_ADD_CHANGEObject. The e-mail message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 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 typically 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 called UCCnet_processWorklist_CATALOGUE_ITEM_CONFIRMATIONObject, based on the Notify_by_eMail collaboration template. This collaboration object sends an e-mail 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. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information.
- It logs the event in the audit_log table. See the section "Using the audit_log table" on page 41 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 e-mail to selected recipients. The e-mail message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 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 called UCCnet_processWorklist_DEAD_LETTER_PUB_RECEIPTObject, which sends an e-mail to selected recipient(s). The e-mail message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 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 completes the following steps:

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 41 for more information about 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 43 for more information about 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 9. 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 completes the following steps:

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 ItemPublicationAdd or ItemPublicationChange request over the *ITEM_ADD_CHANGE* port to the AS2 channel connector.
4. Logs the notification in the audit_log table. See the section "Using the audit_log table" on page 41 for more information about 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 9 and "ItemPublicationChange subflow 1: making updated item information available to trading partners" on page 17.

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 completes the following steps:

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 40 for more information about 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 43 for more information about 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 41 for more information about 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 11 and “ItemPublicationChange subflow 2: processing trading partners’ responses to updated item information” on page 19.

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 41 for more information about 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 Notify_by_eMail collaboration instance to send a notification e-mail to the supplier, 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 in 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.

After 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.

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.

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.

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 27 and "CatalogueItemNotification_Change and CatalogueItemPublication_Change workflow: making updated item information available to trading partners and processing their responses" on page 33. 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 e-mail to selected recipients with the contents of the received message. The e-mail message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 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 called UCCnet_processWorklist_SIMPLE_RESPONSEObject, which sends an e-mail to selected recipient(s). The e-mail message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 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 UCCnet_processWorklist_UNKNOWN_MESSAGESObject, which sends an e-mail to selected recipient(s). The e-mail message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 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 called UCCnet_processWorklist_UNKNOWN_RESPONSEObject, which sends an e-mail to selected recipient(s). The e-mail message, subject, and recipients are configured in this collaboration object's EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. See the section "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52 for more information.

Sending e-mail

UCCnet_ItemSync and UCCnet_processWorklist collaboration objects can be configured to send e-mail 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 e-mail to configured recipients when specific processing circumstances occur.

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

- "Alerting e-mail recipients of processing errors"
- "Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams" on page 52

Alerting e-mail recipients of processing errors

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

UCCnet_ItemSync collaboration object

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

- SEND_EMAIL: This property controls whether e-mail is sent to the e-mail address specified in the SEND_EMAIL_TO configuration property. Set the property value to `all` to send e-mail or to `none` to not send e-mail. If the value is left empty, no e-mail is sent even if recipients exist in the SEND_EMAIL_TO property.
- SEND_EMAIL_TO: This property defines the e-mail addresses where 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 e-mail is sent and to identify the e-mail recipients. The

SEND_EMAIL_TO property defines the e-mail addresses where 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 e-mail. If the property is left blank, the object does not send e-mail. This property must be configured by the user.

Sending e-mail through UCCnet_processWorklist collaboration object subdiagrams

A UCCnet_processWorklist collaboration object contains several subdiagrams that respond to specific types of workflow processing. Several of these subdiagrams include functionality that sends an e-mail to a set of configured addresses by instantiating a collaboration object based on the Notify_by_eMail collaboration template, as follows:

CATEGORY_ADD_CHANGE subdiagram
Instantiates a collaboration object called UCCnet_processWorklist_CATEGORY_ADD_CHANGEObject.

AUTHORIZATION_RESPONSES subdiagram
Instantiates a collaboration object called UCCnet_processWorklist_AUTHORIZATION_RESPONSESObject.

DEAD_LETTER_PUB_RECEIPT subdiagram
Instantiates a collaboration object called UCCnet_processWorklist_DEAD_LETTER_PUB_RECEIPTObject.

CATALOGUE_ITEM_CONFIRMATION subdiagram
Instantiates a collaboration object called UCCnet_processWorklist_CATALOGUE_ITEM_CONFIRMATIONObject

SIMPLE_RESPONSE subdiagram
Instantiates a collaboration object called UCCnet_processWorklist_SIMPLE_RESPONSEObject.

UNKNOWN_MESSAGES subdiagram
Instantiates a collaboration object called UCCnet_processWorklist_UNKNOWN_MESSAGESObject.

UNKNOWN_RESPONSE subdiagram
Instantiates a collaboration object called UCCnet_processWorklist_UNKNOWN_RESPONSEObject.

Each of these objects based on the Notify_by_eMail collaboration template can be configured to contain the e-mail message, subject, and recipients specific to its processing situation through its EMAIL_MESSAGE, EMAIL_SUBJECT, and EMAIL_NOTIFICATION_RCPTS configuration properties. 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 e-mail through the use of a comma-delimited list. Plus, e-mail 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 about these features:

- "Specifying message text, subjects, and recipients in external files" on page 53
- "Specifying changing individual or multiple message recipients" on page 53
- "Using substitution variables in message and subject text" on page 53

The value of the AUTO_RESPOND property of the UCCnet_processWorklist collaboration object determines whether e-mail 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 e-mail message text, subject text, and recipients to contain the names of files. These files contain the actual e-mail 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 e-mail 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 file name 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 file name. 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 file name containing the e-mail 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 e-mail value directly from the attribute.

Specifying changing individual or multiple message recipients

A Notify_by_eMail collaboration object allows all e-mail messages to be routed to an administrator or to a specific role in an organization (such as a Category Manager), without the need to maintain the e-mail recipient's fully qualified e-mail address in every collaboration object that might send e-mail. By placing the e-mail 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 e-mail 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

E-mail 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 e-mail message and subject text must appear as:

```
${variable_name}
```

Note: These characters might need 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 IBM WebSphere Business Integration Express for Item Synchronization product. 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 IBM WebSphere Business Integration Express for Item Synchronization product 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 IBM WebSphere Business Integration Express for Item Synchronization product. 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

Notices

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

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

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

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.

WebSphere Business Integration Express for Item Synchronization V4.3.1

WebSphere Business Integration Express for Plus for Item Synchronization V4.3.1