SeeBeyond ICAN Suite

UN/EDIFACT Manager Composite Application User's Guide

Release 5.0

Version 20050408174322.

Contents

Chapter 6

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces you to this guide, its general purpose and scope, and its organization. It also provides sources of related documentation and information.

What's in This Chapter

1.1 About This Document

This user's guide describes how to install and use the UN/EDIFACT Manager Composite Application (UN/EDIFACT Manager) to create ICAN Projects that process and validate UN/EDIFACT messages.

1.1.1 What's in This Document

This guide includes the following chapters:

- **Chapter 1, "Introduction"** provides an overview of this document's purpose, contents, writing conventions, and supported documents.
- **Chapter 2, "Overview of the UN/EDIFACT Manager"** provides an overview of the UN/EDIFACT Manager Composite Application.
- **Chapter 3, "Installing the UN/EDIFACT Manager"** describes how to install the UN/EDIFACT Composite Application and sample Projects. It also lists system requirements and supported operating systems and external applications.
- **Chapter 4, "Working with Validation BPs"** describes how to customize validation handler BPs by using an example of locating and modifying a SeeBeyond-supplied B2B protocol.
- **Chapter 5, "Configuring Trading Partners"** discusses the UN/EDIFACT-specific items that can or must be configured in eXchange Partner Manager (ePM).

- **Chapter 6, "Working with the UN/EDIFACT Sample"** describes how to import, configure, and run the UN/EDIFACT sample.
- **Appendix A, "OTD Syntax Validation BPs"** provides in-depth technical information on B2B protocol processes for handling Object Type Definition (OTD) syntax validation.

1.1.2 Intended Audience

This user's guide is intended for ICAN Project developers who have experience with he UN/EDIFACT protocol standards.

1.1.3 Document Conventions

The following conventions are observed throughout this document.

Table 1 Document Conventions

Text	Convention	Example
Names of buttons, files, icons, parameters, variables, methods, menus, and objects	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments, code samples	`Fixed` font. Variables are shown in ***`bold italic`***.	`bootstrap -p` ***`password`***
Hypertext links	**Blue** text	See **Related Documents** on page 9
Hypertext links for Web addresses (URLs) or email addresses	**Blue underlined** text	**http://www.seebeyond.com** **docfeedback@seebeyond.com**

1.1.4 Screenshots

Depending on what products you have installed, and how they are configured, the screenshots in this document may differ from what you see on your system.

1.2 Related Documents

For more information about eGate Integrator, eInsight Partner Manager, eXchange Integrator, and the eWays used for the UN/EDIFACT sample Projects, refer to the following documents:

- *SeeBeyond ICAN Suite Installation Guide*
- *eGate Integrator User's Guide*

- *eGate Integrator JMS Reference Guide*
- *eGate Integrator System Administrator Guide*
- *eGate Integrator Deployment Guide*
- *eXchange Integrator User's Guide*
- *eXchange Integrator Designer's Guide*
- *eInsight Business Process Manager User's Guide*
- *UN/EDIFACT OTD Library User's Guide*
- *Oracle eWay Intelligent Adapter User's Guide*
- *Batch eWay Intelligent Adapter User's Guide*
- *File eWay Intelligent Adapter User's Guide*

1.3 References

The following resources provide additional information about the UN/EDIFACT protocol:

- The United Nations Economic Commission of Europe (UN/ECE) is one of the five regional commissions of the United Nations. The UN/ECE Web site contains technical information concerning rules, standards, recent UN/EDIFACT directories, syntax, and so on.

  http://www.unece.org/trade/untdid/welcome.htm

- UN/EDIFACT publishes the messages for each version separately from the envelopes (header and trailer segments) that are used with those messages.

  The messages are published at:

  http://www.gefeg.com/en/standard/edifact/index.htm

  The envelopes are published at:

  http://www.gefeg.com/jswg/

1.4 SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

1.5 SeeBeyond Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

docfeedback@seebeyond.com

Chapter 2

Overview of the UN/EDIFACT Manager

This chapter provides a general overview of the UN/EDIFACT Manager and its place in the ICAN Suite.

What's in This Chapter

2.1 About the UN/EDIFACT Manager Composite Application

Note: *This release of the UN/EDIFACT Manager supports EDIFACT Batch only.*

The UN/EDIFACT Manager Composite Application integrates with eGate Integrator, eXchange Integrator, and the UN/EDIFACT OTD Library to enable you to design ICAN Projects that process and validate UN/EDIFACT messages.

The UN/EDIFACT Manager includes a *business service*—a sequence of events incorporating the rules set by the protocol specifications, such as:

- Interchange and acknowledgment processing
- Business message correlation
- Enveloping and de-enveloping
- Document batching and splitting
- Event archiving

UN/EDIFACT Manager and eXchange Integrator

eGate Integrator and eXchange Integrator enable you to build ICAN Projects that process standard B2B business protocols and enveloping protocols such as UN/EDIFACT. The UN/EDIFACT Manager works with eXchange to provide the following during message processing:

- Error handling
- Message tracking
- Trading partner profile database lookup

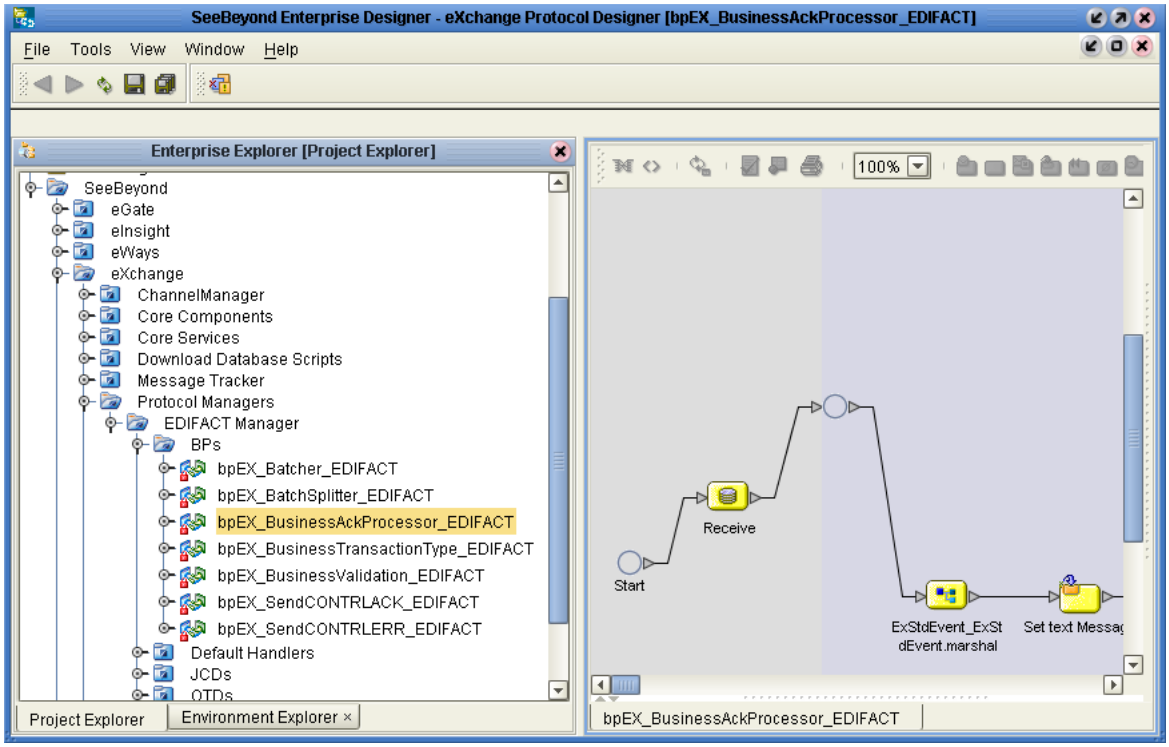
UN/EDIFACT Manager and the UN/EDIFACT OTD Library

The UN/EDIFACT Manager provides packaged Business Protocol (BP) rules to validate UN/EDIFACT message structures, which are called Object Type Definitions (OTDs) in the ICAN Suite. The ICAN Suite provides packaged UN/EDIFACT OTDs with the UN/EDIFACT OTD Library. You can also build your own OTDs with the SEF OTD wizard, which is supplied with eGate Integrator.

Importing and customizing validation rules

The UN/EDIFACT Manager enables you to tailor validation rules for your specific Project; you can import validation rules into your ICAN Project and customize them. For information about importing validation rules, refer to **"Importing Validation BPs into Projects" on page 31**.

The figure below shows the BPs supplied with the UN/EDIFACT Manager and the canvas shows one of the UN/EDIFACT BPs. All business logic is exposed, and each action is customizable.



2.2 How the UN/EDIFACT Manager Works

In the View layer: The eXchange Protocol Designer provides the ability to create UN/EDIFACT Projects, and the eXchange Message Tracker allows searching and viewing of UN/EDIFACT messages.

In the Services Orchestration layer: You use the eXchange Service Designer to design a transaction set; then the UN/EDIFACT Project prepares and returns the interchange and functional acknowledgments to the trading partner, and also performs message correlation to associate the business response to the request.

In the Integration Services layer: The UN/EDIFACT Project validates UNB and UNG envelopes from incoming messages, prepares UNB and UNG envelopes for outgoing messages, batches together documents to be delivered as a single transaction (UNB), and records the activity in Message Tracking.

2.3 About the UN/EDIFACT Protocol

The United Nations/Electronic Data Interchange (UN/EDIFACT) for Administration, Commerce and Transport protocol was developed for the electronic exchange of machine-readable information between businesses.

The UN/EDIFACT Working Group (EWG) develops, maintains, interprets, and promotes the use of the UN/EDIFACT standard.

UN/EDIFACT messages are structured according to very strict rules. Messages are in ASCII format. The standard defines all these message elements, their sequence, and also their grouping.

UN/EDIFACT publishes the messages for each version separately from the envelopes (header and trailer segments) that are used with those messages.

The messages are available online at:

http://www.gefeg.com/en/standard/edifact/edifact.htm

The envelopes are available online at:

http://www.gefeg.com/jswg/

A new version of UN/EDIFACT messages is released several times a year, containing most of the messages in the previous version, plus any new messages that have been approved by the standards organization. The envelopes are updated with a new version infrequently.

2.3.1 Message Structure

UN/EDIFACT messages have a message structure, which indicates how data elements are organized and related to each other for a particular EDI transaction. In the ICAN Suite, message structures are defined as OTDs. Each OTD consists of the following:

- Physical hierarchy

The predefined way in which envelopes, segments, and data elements are organized to describe a particular UN/EDIFACT EDI transaction.

- Delimiters

  The specific predefined characters that are used to mark the beginning and end of envelopes, segments, and data elements.

- Properties

  The characteristics of a data element, such as the length of each element, default values, and indicators that specify attributes of a data element—for example, whether it is required, optional, or repeating.

The message level structure of an invoice that is sent from one trading partner to another defines the header, trailer, segments, and data elements required by invoice transactions. The UN/EDIFACT OTD for a specific version includes message level structures for each of the transactions available in that version. You can use these structures as provided, or customize them to suit your business needs.

eXchange uses OTDs based on UN/EDIFACT message structures to verify that the data in the messages coming in or going out is in the correct format. There is a message structure for each UN/EDIFACT transaction. The list of transactions provided is different for each version of UN/EDIFACT.

2.3.2 Segment Table

A key section for each document appears at the end of each message. Here you can find a segment table displaying the message structure which shows the order of segments and the manner in which they repeat. See the example of an APERAK transaction below.

Note: *For information on specific messages, see the United Nations Web site and view the message type by code. The URL is:*

http://www.unece.org/trade/untdid/

```
4.3.1 Segment table

Pos   Tag Name                                    S  R

0010  UNH Message header                          M  1
0020  BGM Beginning of message                    M  1
0030  DTM Date/time/period                        C  9
0040  FTX Free text                               C  9
0050  CNT Control total                           C  9

0060   ÄÄÄÄÄ Segment group 1  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ C  99ÄÄÄÄÄÄÄÄÄÄ¿
0070  DOC Document/message details                M  1           ³
0080  DTM Date/time/period                        C  99ÄÄÄÄÄÄÄÄÄÄÙ

0090   ÄÄÄÄÄ Segment group 2  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ C  9ÄÄÄÄÄÄÄÄÄÄ¿
0100  RFF Reference                               M  1           ³
0110  DTM Date/time/period                        C  9ÄÄÄÄÄÄÄÄÄÄÙ

0120   ÄÄÄÄÄ Segment group 3  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ C  9ÄÄÄÄÄÄÄÄÄÄ¿
0130  NAD Name and address                        M  1           ³
0140  CTA Contact information                     C  9           ³
0150  COM Communication contact                   C  9ÄÄÄÄÄÄÄÄÄÄÙ

0160   ÄÄÄÄÄ Segment group 4  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ C  99999ÄÄÄÄÄÄÄ¿
0170  ERC Application error information           M  1           ³
0180  FTX Free text                               C  1           ³
                                                                 ³
0190   ÄÄÄÄÄ Segment group 5  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ C  9ÄÄÄÄÄÄÄÄÄ¿³
0200  RFF Reference                               M  1           ³³
0210  FTX Free text                               C  9ÄÄÄÄÄÄÄÄÄÄÁÙ
0220  UNT Message trailer                         M  1
```

The “S” column indicates whether the loop or segment is “M” (mandatory) or “C” (conditional).

The “R” column indicates the maximum number of repetitions of the segment or loop.

The “Ä“(A-umlaut) can be interpreted as a horizontal line; it indicates the first and last segments in a loop.

The “¿,” “ ³,” and “Ù” characters at the end of some lines indicate the first, continuing, and last segments of a loop respectively. Where there are more than one “ ³” at the end of a line, there are nested loops.

2.3.3 Loops

A loop consists of two or more data segments that contain a block of information (for example: company name, street address, mailing address, city, state, and zip code) that can repeat multiple times.

- Locate the fields by specifying:
  - Transaction set (for example, APERAK)
  - Loop (for example, segment group 1)
  - Which occurrence of the loop

- Segment (for example, DOC)
- Field number (for example, DOC00)
- Which occurrence of the segment (if repeating)

2.3.4 Envelopes

UN/EDIFACT publishes the envelope segments in the separate syntax document with independent version numbers. For example, there is a version 3 and a version 4, either of which can be used with any version of the messages. v3 and v4 are two separate interchange envelope syntax versions, and do not dictate the message modes. v3 is outdated and only handles batch messages, whereas v4 can handle batch or interactive messages. This version of the UN/EDIFACT Manager supports batch messages only.

Table 2 UN/EDIFACT Envelopes

ENVELOPE	UN/EDIFACT Batch Messages	
	start	**end**
Interchange Envelope	UNA/UNB	UNZ
Functional Group Envelope	UNG	UNE
Message Envelope	UNH	UNT

UNA segment

All UN/EDIFACT OTDs have a UNA segment, but these segments are optional. The UNA segment is used to indicate unusual delimiter characters.

The string has a mandatory fixed length of 9 characters. The first three are "UNA," immediately followed by the 6 characters as defined in ISO 9735.

The UNA segment has six elements with the node name of EUNA*?*_*?*_*XXX* where *?* is a position number (1 to 6) and *XXX* contains the descriptive information for that element node.

Control messages

Control messages (versus business messages) are also published separately with the syntax document. There is a **CONTRL** message for both v3 and v4 batch envelopes only.

Each version of the UN/EDIFACT OTDs include both a v3 **CONTRL** and a v4 **CONTRL** message.

2.3.5 Delimiters

The delimiters are dynamically set.

2.3.6 Security

EDI-INT is an international standard for secure EDI transmissions. It is emerging as a widespread EDI security standard. It has the following features:

- Uses HTTP and PKI
- MIME and public key cryptography
- Many options

Note: *This is only related to the data transmission and not to the parsing of the UN/EDIFACT message itself.*

For additional information:

http://www.ietf.cnri.reston.va.us/ids.by.wg/ediint.html

2.4 Examples of EDI Usage

This section provides an overview of EDI payment processing, followed by a description of the types of EDI transactions, then examples of credit transfer scenarios.

Note: *This is just an example of how UN/EDIFACT and payments processing is used. Not everything said here applies to all UN/EDIFACT messages.*

2.4.1 Overview of EDI Payments Processing

EDI payments processing is a combination of collections and disbursements, with the processing taking the form of debits and credits. It can also include a related bank balance, as well as transaction and account analysis reporting mechanisms.

Most of the other EDI trading partner communications are handled either directly between the parties or indirectly through their respective value-added networks (VANs).

Making an electronic payment requires a financial intermediary, usually the bank or banks that hold deposit accounts of the two parties.

Exchange of remittance information

EDI involves the exchange of remittance information along with the order to pay. In the United States this can become complex as two standards are involved in the transaction. Think of the remittance information as an electronic check stub, which can follow one of the following paths to complete the transaction:

- Directly between trading partners or through their respective EDI VAN mailboxes
- Through the banking system, with the beneficiary receiving notice from his bank
- By the originator to the originator's bank as an order to pay, which in turn reports to the beneficiary

Routing of remittance information

The trading partners and the capabilities of their respective banks determine the routing of the electronic check stub, and whether the payment is a debit authorized by the payor and originated by the beneficiary or a credit transfer originated by the payor.

Other opportunities to exchange information between a bank and its customer include:

- Daily reports of balances and transactions
- Reports of lockbox and electronic funds transfer (EFT) remittances received by the bank
- Authorizations issued to the bank to honor debit transfers
- Monthly customer account analysis statements
- Account reconcilement statements
- Statements of the demand deposit account

Exchange of payment orders

A subset of EDI, the electronic payment mechanism activates the exchange of payment orders; value transfers from one account to another, including the related remittance information in standardized machine-processable formats. The electronic payment can be either:

- Credit transfer, initiated by the payor

  or

- Debit transfer, initiated by the payee as authorized by the payor

Regardless of how the credit transfer was initiated, the payor sends a payment order to its bank as an UN/EDIFACT PAYEXT message.

The bank then adds data in a format prescribed in the United States by the National Automated Clearing House Association (NACHA) and originates the payment through the Automated Clearing House (ACH) system.

Functions a payment must perform

A corporate-to-corporate payment *must* perform two functions:

- Transfer value
- Move remittance data from the payor to the payee

When a credit transfer occurs, the mandatory functions raise the issue of how the funds and remittance information will travel, which is either:

- Together through the banking system

  or

- Separated and traveling by different routes

Formats for transporting a payment

The UN/EDIFACT PAYEXT is a data format for transporting a payment order from the originator to its bank. This payment order is either an:

- Instruction to the originator's bank to originate a credit transfer

  or

- Instruction to its trading partner to originate a debit transfer against the payor's bank account

Once this decision has been made, the PAYEXT transports the remittance information to the beneficiary. As stated above, this transfer can either be through the banking system or via a route that is separate from the transport of funds.

Note: *Whenever the PAYEXT remittance information is not transferred with the funds, the PAYEXT (information only) can be transmitted directly from the originator to the beneficiary. It can also be transmitted through an intermediary, such as a VAN.*

Issuance of a payment order

Before funds can be applied against an open accounts receivable, the beneficiary must reconcile the two streams—the payment advice from the receiving bank and the remittance information received through a separate channel—which were separated during the transfer. If this reconciliation does not take place and if the amount of funds received differs from the amount indicated in the remittance advice, the beneficiary's accounts receivable ledger will suffer from a multitude of problems.

The value transfer begins when the originator issues a payment order to the originator's bank. If a credit transfer is specified, the originator's bank charges the originator's bank account and pays the set sum to the beneficiary's bank for credit to the account of the beneficiary.

The originator becomes the same party as the beneficiary when the payment order specifies a debit transfer. When this happens, the beneficiary's bank originates the value transfer, the payor's account is debited (charged) for a set amount, which is credited to the originator's (beneficiary's) bank account. Either prior to or concurrent with a presentment of a debit transfer, the payor must issue approval to its bank to honor the debit transfer. This debit authorization or approval can take one of the following forms:

- Individual item approval

- Blanket approval of all incoming debits with an upper-dollar limit
- Blanket approval for a particular trading partner to originate any debit
- Some combination of the above

2.4.2 Payment-Related EDI Transactions

UN/EDIFACT routes the Payment Order/Remittance Advice from the originator to the beneficiary in a point-to-point method.

UN/EDIFACT uses different messages for each of these point-to-point transmissions, and separates the banking (Payment Order) function from the financial (Remittance Advice) function. This, in effect, creates the following distinct functions:

- The originator company uses the Payment Order to notify its bank that a funds transfer should take place (PAYEXT, PAYMUL).
- The originator company uses the Remittance Advice to notify the beneficiary of the payment (REMADV from originator, CREADV/DEBADV to beneficiary).
- The EFT actually moves the monetary value from one bank to another bank (ACH in the United States; SWIFT or CHIPS in Europe). For an UN/EDIFACT Payment Order/Remittance Advice, see Table 3 below. Table 4 lists other related UN/EDIFACT transactions.

Table 3 UN/EDIFACT Payment Order/Remittance Advice

Step	Description of Action	UN/EDIFACT	Step
1	Payment Order from originator to its bank	PAYEXT, PAYMUL	1
1	Remittance Advice from originator to be passed on to beneficiary	REMADV	2
2	Remittance advice to beneficiary	CREADV, DEBADV	2
3	EFT between banks	SWIFT, CHIPS in Europe; ACH in the U.S.	3

Table 4 Other Related Transactions

Transaction	UN/EDIFACT
Debit Authorization	AUTHOR
Payment Cancellation Request	FINCAN
Application Control Totals	(none)

2.4.3 Understanding Enveloping Scenarios

We will use two credit transfer scenarios to give you a better understanding of the addressing issue:

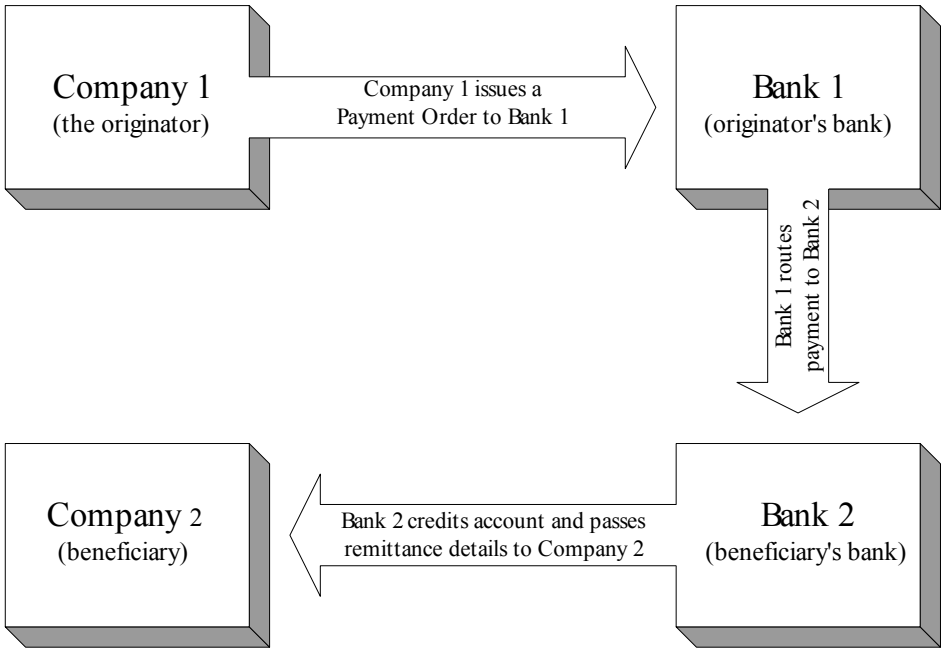
- Point-to-point
- End-to-end

These scenarios involve two corporate trading partners and their respective banks:

- Company 1
- Company 2
- Bank 1
- Bank 2

Company 1 (the originator) issues a Payment Order (credit transfer) through Bank 1, which in turn routes the payment through the ACH to Bank 2 (the beneficiary's bank). Bank 2 then credits the account and passes the remittance details to its customer, Company 2 (the beneficiary).

Figure 1 Example Payment Scenario



Although trading partners prefer to consider this payment mechanism as an end-to-end operation, the banking system's mechanism is actually a series of point-to-point transactions, mainly:

- From the originator to the originator's bank
- From the originator's bank to the beneficiary's bank
- From the beneficiary's bank to the beneficiary

In the PAYEXT, the identity of the originator, the originator's bank, the beneficiary's bank, and the beneficiary are established in the header of the transaction set (message) itself. Table 5, below, shows UN/EDIFACT headers.

Table 5 Sample UN/EDIFACT Headers

Envelope	UN/EDIFACT start	UN/EDIFACT end
Interchange Envelope	UNA/ UNB	UNZ
Functional Group Envelope	UNG	UNE
Message Envelope	UNH	UNT

Point-to-Point Scenario

In our example point-to-point scenario, the steps are as follows:

1. Company 1 sends a payment file to Bank 1 using a UNB interchange header in which:
   - Sender ID = Company 1
   - Receiver ID = Bank 1
2. Bank 1 replaces Company 1's UNB with its own UNB as follows:
   - Sender ID = Bank 1
   - Receiver ID = Bank 2
3. Bank 2 receives the payment file, and creates a new UNB to send the contents to Company 2 that shows:
   - Sender ID = Bank B
   - Receiver ID = Company B

Because the interchange control header (UNB) changes at each point, it is important that the functional group header (UNG) is not changed.

The UN/EDIFACT functional group envelope (UNG & UNE) is optional; the UNG and UNE segments can be both be missing, or both be present.

Maintaining the functional group guarantees that the payment file retains the Company 2 information. As some originators do not care what happens to the original UNB, it is imperative that each bank in the chain ensure that the UNB contains PAYEXTs that are destined for only one Company 2. This rule makes it so that the banks only have to look at the UNB for addressing information, and the receiving company can respond with a Functional Acknowledgment (CONTL) to the originator.

End-to-end scenario

In our example end-to-end scenario, the steps are:

1. Company 1 sends a payment file to Bank 1 using a UNB in which:

- Sender ID = Company 1
- Receiver ID = Company 2

2 Bank 1 does not disturb the UNB, which continues to show:

- Sender ID = Company 1
- Receiver ID = Company 2

3 Bank 2 does not disturb the UNB, which continues to show:

- Sender ID = Company 1
- Receiver ID = Company 2

Note: *Banks usually recommend end-to-end scenarios.*

In this scenario, both the originator and the beneficiary's bank are prohibited from altering the UNB/UNZ interchange envelope information. This makes it mandatory for the originating company to create an UNB envelope, and a separate transmission, for each destination end point. Unfortunately, this could potentially mean hundreds of such end points in each accounts payable cycle.

SeeBeyond's EDI enveloping features in the eXchange product automatically remove both the interchange and functional group envelopes and re-create the point-to-point envelopes. Special handling is required to override this default.

2.4.4 Payment Acknowledgments

The acknowledgment of the receipt of a payment order is an important issue. Most corporate originators want to receive at least a Functional Acknowledgment (CONTRL) from the beneficiary of the payment. The CONTRL is created using the data about the identity and address of the originator found in the UNB and/or UNG segments. In UN/EDIFACT, CONTRL is a point-to-point acknowledgment.

Table 6 Types of UN/EDIFACT Acknowledgments

UN/EDIFACT	Envelope
CONTRL	System Level Acknowledgment (receipt)
CONTRL	Function Acknowledgment (point-to-point)
	Application Advice (end-to-end)

2.5 UN/EDIFACT Directory Support

The UN/EDIFACT Manager Composite Application provides OTDs for the following UN/EDIFACT directories:

- D.01A and B
- D.00A and B

- D.99A and B
- D.98A and B
- D.97A and B
- D.96A and B
- D.95A and B

2.6 SEF File Support

You can use this product with custom SEF OTDs built with the SEF OTD wizard. The wizard supports SEF versions 1.5 and 1.6.

The SEF OTD wizard does not handle the following information and sections:

- In the .SEMREFS section, semantic rules with its type of the “exit routine” are ignored as per SEF specification. An exit routine specifies an external routine (such as a COM-enabled server program supporting OLE automation) to run for translators or EDI data analyzers.
- The .TEXT sections (including subsections such as .TEXT,SETS, .TEXT,SEGS, .TEXT,COMS, and .TEXT,ELMS) are ignored, because these sections store information about changes in a standard's text, such as notes, comments, names, purposes, descriptions, titles, semantic notes, explanations, and definitions.

Chapter 3

Installing the UN/EDIFACT Manager

This chapter describes how to install the UN/EDIFACT Manager Composite Application, its documentation, and sample Projects. This chapter also includes the system requirements and supported operating systems for the UN/EDIFACT Manager Composite Application.

What's in This Chapter

3.1 System Requirements

Each UN/EDIFACT validation BP **.sar** file requires approximately 5 MB disk space. The large size of the UN/EDIFACT validation BPs normally requires an increased heap size property of the Enterprise Designer. For information, refer to **"Increasing the Enterprise Designer Heap Size" on page 29**.

Other than that, the system requirements for the UN/EDIFACT Manager are the same as those for eGate Integrator and eInsight Business Process Manager. For information, refer to the *SeeBeyond ICAN Suite Installation Guide*.

3.2 Supported Operating Systems

The UN/EDIFACT Manager Composite Application is available for the following operating systems:

- Microsoft Windows 2000 SP3 or SP4, Windows XP SP1a, and Windows Server 2003
- Sun Solaris 8 and Solaris 9, with required patches

- HP Tru64 V5.1A, with required patches
- HP-UX 11.0, 11i (PA-RISC), and 11i v2.0 (11.23) with required patches and parameter changes
- IBM AIX 5.1L and AIX 5.2 (either 64-bit kernel or 32-bit kernel with 64-bit extension), with required maintenance level patches
- Red Hat Linux 8 (Intel *x*86) and Linux Advanced Server 2.1 (Intel *x*86)

3.3 Supported External Applications

Database support for the UN/EDIFACT Manager is the same as for eXchange Integrator. For information, refer to the *eXchange Integrator User's Guide*.

3.4 Required ICAN Suite Products

The UN/EDIFACT Manager Composite Application requires the following products to be installed:

- eGate Integrator
- Batch eWay Intelligent Adapter
- Oracle eWay Intelligent Adapter
- eXchange Integrator

3.5 Installing the UN/EDIFACT Composite Application

During the UN/EDIFACT Manager installation process, the Enterprise Manager, a Web-based application, is used to select and upload products as **.sar** files from the ICAN Suite installation CD-ROM to the Repository.

The installation process includes the following steps:

- Installing the Repository
- Uploading products to the Repository
- Downloading components (such as Enterprise Designer and Logical Host)
- Viewing product information home pages

Follow the instructions for installing the eGate Integrator in the *SeeBeyond ICAN Suite Installation Guide*, and include the steps below to install the UN/EDIFACT Manager. You must have uploaded a **license.sar** to the ICAN Repository that includes a license for the UN/EDIFACT Manager.

To install the UN/EDIFACT Manager Composite Application

1. Upload the following files to the eGate Repository using the Enterprise Manager as described in the *SeeBeyond ICAN Suite Installation Guide*. You must upload the **.sar** files in the order presented here:
   - A **eGate.sar**
   - B **BatcheWay.sar** and **Oracle.sar** (must be installed before you install eXchange)
   - C **eXchange.sar**
   - D **EDIFACT_Manager.sar** (to install UN/EDIFACT Manager)
   - E **UN_EDIFACT_OTD_Lib_***.sar** (to install the EDIFACT OTDs as described in the *UN/EDIFACT OTD Library User's Guide*)
   - F **UN_EDIFACT_OTD_Validation_BP_***.sar** (to install the BPs to validate the OTDs installed in step D—optional)
2. If you need to build custom SEF OTDs, upload **SEF_OTD_Wizard.sar** from Products CD3.
3. To work with the sample Projects as described in **"Working with the UN/EDIFACT Sample" on page 38**, do the following:
   - A Upload the following items:
     - **UN_EDIFACT_OTD_Lib_v3_D00B.sar** (to install the UN/EDIFACT OTDs)
     - **FileeWay.sar** (to install the File eWay)
     - **HTTPeWay.sar** (to install the HTTP(S) eWay)
     - **EDIFACT_ManagerDocs.sar** (to install the user's guide and the sample)
   - B In the Enterprise Manager, click the **DOCUMENTATION** page, and click **UN/EDIFACT Manager Composite Application.**
   - C In the right-hand pane, click **Download Sample**, and save the **.zip** file to **c:\temp\eXchange**.
4. Start (or restart) the Enterprise Designer, and click **Update Center** on the **Tools** menu. The Update Center shows a list of components ready for updating.
5. Click **Add All** (the button with a doubled chevron pointing to the right). All components move from the **Available/New** pane to the **Include in Install** pane.
6. Click **Next** and, in the next window, click **Accept** to accept the license agreement.
7. When the progress bars indicate the download has ended, click **Next**.
8. Review the certificates and installed modules, and then click **Finish**.
9. When prompted to restart Enterprise Designer, click **OK**.

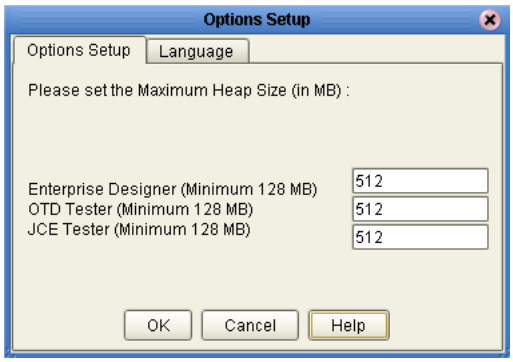
3.6 Increasing the Enterprise Designer Heap Size

Due to the size of the UN/EDIFACT OTDs, you may need to increase the heap size property of the Enterprise Designer. If the heap size is not increased, out of memory errors may occur.

To increase the Enterprise Designer heap size

1. On the **Tools** menu in Enterprise Designer, click **Options**. The **Options Setup** dialog box appears.
2. Set the configured heap size for the Enterprise Designer, OTD Tester, and JCE Tester to no less than 512 MB, and click **OK**.

Figure 2 Increasing Enterprise Designer Heap Size



3. Restart Enterprise Designer.

3.6.1 Resolving Memory Errors at Enterprise Designer Startup

If an out of memory error occurs at Enterprise Designer startup, change the setting in the **heapSize.bat** file. This file resides in the folder ***ICAN_Suite*\edesigner\bin**, where *ICAN_Suite* is the folder where eGate Integrator is installed.

Open the file with a text editor, and increase the heap size setting. Save the file, and restart the Enterprise Designer.

3.7 Configuring the Oracle Database

Before using the UN/EDIFACT Manager, refer to the *eXchange Integrator User's Guide* and the *Oracle eWay Intelligent Adapter User's Guide* for additional installation instructions for setting up the Oracle database. eXchange provides the **createuser.sql** script to create username/password combinations and the **createdb.sql** script to populate tablespaces for each user.

To use the sample Projects provided with the UN/EDIFACT Manager, the database must contain the following usernames and passwords:

- user: **ex_A**, password: **ex_A**

- user: **ex_B**, password: **ex_B**

Chapter 4

Working with Validation BPs

The UN/EDIFACT Manager provides packaged BPs that validate corresponding UN/EDIFACT message structures (OTDs). This chapter describes how you can import validation BPs into ICAN Projects, and how you can customize the BPs.

What's in This Chapter

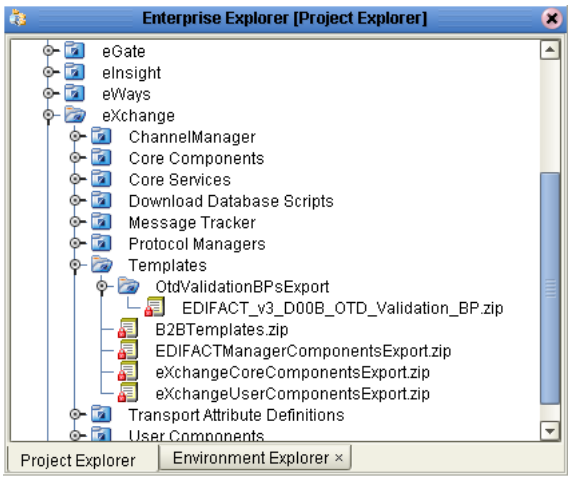
4.1 Importing Validation BPs into Projects

When you install a validation BP as described in **Installing the UN/EDIFACT Composite Application** on page 27, the BP is included in Enterprise Designer as a **.zip** file. This enables you to save the .zip file and import this .zip file into your ICAN Project as described below.

To import validation BPs

1 In the **Project Explorer** tab of Enterprise Designer, expand the **SeeBeyond > eXchange > Templates > OTDValidationBPsExport** folders.

Figure 3 Validation BP **.zip** Files



2 Right-click the **.zip** file and click **Export**.

3 In the **Save** dialog box, select the location for the **.zip** to be saved and click **Save**.

4 In the **Project Explorer** tab of Enterprise Designer, right-click the Project to use the validation rules, and click **Import**.

5 In the **Import Manager** dialog box, navigate to the validation BP **.zip** file, double-click the file, and click **Import**.

6 Click **OK** at the confirmation message and click **Close** to close the Import Manager dialog box.

The validation BP now displays under the ICAN Project.

Chapter 5

Configuring Trading Partners

This chapter describes how to configure Trading Partners in eXchange Partner Manager (ePM).

What's in This Chapter

5.1 Overview of the ePM Parameter Hierarchy

A trading partner's configuration parameters can be set at three different levels:

- You can configure the properties and components of the ***trading partner*** (TP) itself.
- You can configure the properties of ***trading partner profiles*** (TPPs) within a TP.
- You can configure the properties and business actions for ***services*** defined for a TPP. Business and messaging services are organized according to Business Attributes Definitions (BAD).

5.2 Configuring Trading Partners

Configuration parameters for the Trading Partner itself are organized into two major tabs: **Properties** and **Components**.

5.2.1 Configuring Trading Partners

The top of the **Delivery Channels** subtab lists all bindings currently defined for the TP's external delivery channels (XDCs). An XDC "binding" is an association of XDC metadata parameters to a particular set of values.

The **Enveloping Channels** subtab allows you to add, modify, and delete bindings for the TP's enveloping channels. The UN/EDIFACT protocol, unlike AS2 and ebXML, requires enveloping channels.

To configure Trading Partners

1 In the **Properties > General** tab for the Trading Partner, enter the Trading Partner name.

2 Click **Components** and click **Delivery Channels** for the Trading Partner.

3 To specify bindings for delivery channels:

A Click the XDC for which you want to add a new binding from the list of delivery channels defined for the current host.

B To specify general settings, enter the following information in the General tab and click **Save**:

Table 7 General XDC Bindings Settings

Parameter Name	Default Value	Description / Notes
Delivery Channel	(read-only)	This tells you which XDC was chosen when the binding was initially created.
Binding Name	(dropdown list)	Initially, this shows you the name supplied when the binding was created. Do not use spaces in this name.
Description		
Delivery Channel Handler	[None]	If your B2B host has a handler BP of this type, choose it from the dropdown list. (SeeBeyond supplies two standard delivery channel handler BPs: bpEX_DeliveryChannel_File, used in the sample, and bpEX_DeliveryChannel_FTP.)
Delivery Message Syntax Validation Handler	[None]	If your B2B host has a handler BP of this type, select it from the list.

C In the **ToPartner Transport** tab, enter the required values.

The names, types, and possible values for parameters in the **ToPartner Transport** tab depend on the XDC definition in the B2B host's External Delivery Channels window: The transport attributes definition (TAD) that was specified as "To Partner Transport Attributes Definition" for this XDC specifies the attributes that appear in the "ToPartner Transport" tab for all bindings referencing the XDC.

D In the **FromPartner Transport** tab, enter the required values.

The names, types, and possible values for parameters in the **FromPartner Transport** tab depend on the XDC definition in the B2B host's External Delivery Channels window: The TAD that was specified as "From Partner Transport Attributes Definition" for this XDC specifies the attributes that appear in the "FromPartner Transport" tab for all bindings referencing the XDC.

4 To configure enveloping channels:

A Click **Components** and click **Enveloping Channels** for the Trading Partner.

The top of the **Enveloping Channels** subtab lists all bindings currently defined for the TP's enveloping protocols. A "binding" is an association of enveloping protocol metadata parameters to a particular set of values.

B If you have not already done so: In the Explorer (left) pane, select the trading partner; then, in the Trading Partner (right) pane, click the **Components** tab and the **Enveloping Channels** subtab.

C In the Trading Partner (right) pane, click **New**.

D Click the enveloping attributes definitions (EAD) for which you want to add a new binding from the EADs list, and click **Continue**.

In the list of bindings at the top of the pane, a row for the new binding is added to the end; below the list, three new subtabs appear, with **General** preselected. All three subtabs for the new binding have been populated with default parameters; to modify them, see the following procedure.

E In the **General** tab, enter the following information and click **Save**:

Entries marked *req* indicate an entry is required but the default is blank.

Table 8 Setting ToPartner Envelope Binding Configurations

Parameter Name	Default Value	Description / Notes
Release Quantity	1	Nonnegative integer. Required. Specifies the threshold beyond which a batch send is triggered. Set Release Quantity to a very high value (99999999) if you want to send messages on a schedule, using Release Scheduler String.
Release Scheduler String		A expression specifying when and how often to run. The expression uses **cron** syntax and consists of six (or optionally seven) arguments, separated by spaces, to specify: **second, minute, hour, day-of-month, month, day-of-week** (and optionally **year**). When a trigger time occurs, a batch is sent even if its count has not reached Release Quantity.
Time-Out (Minutes)	req	Nonnegative integer. Required. Specifies maximum time to wait for a reply before attempting a re-send.
Max Retry Count	req	Nonnegative integer. Required. Specifies maximum number of times to retry sending before giving up.
Batcher Handler	[None]	If your B2B host has a handler BP of this type, choose it from the list. (SeeBeyond supplies a standard batcher handler BP for UN/EDIFACT: bpEX_Batcher_EDF, used in the sample.)

Table 9 Setting FromPartner DeEnvelope Binding Configurations

Parameter Name	Default Value	Description / Notes
Batch Splitter Handler	[None]	Select a BP from the list.

5.3 Setting Up Trading Partner Profiles (TPP)

Configuration parameters for each TP Profile are all contained in the **Properties** tab.

Table 10 Parameters in the (TP->TPP->) "Properties" Tab

Parameter Name	Default Value	Description / Notes
Profile Name		Use this field to rename a TP profile.
Max Concurrent Conversations	(no preset value)	Use this field to specify an upper limit to the number of simultaneous business services being processed.

5.4 Configuring Business Services

Configuration parameters for a business service (or messaging service) are organized into two major tabs: **Properties** and **Components**.

5.4.1 Business Service > Business Actions

In the Business Service Configuration window, click the Business Actions tab to display a list of all business actions defined for the current business service. You can expand one or more of the business actions to view or modify its parameters; see Table 11.

5.5 Configuring Business Actions

Table 11 Parameters in the (TP->TPP-> Business Service) "Business Actions" Tab

Parameter Name	Default Value	Description / Notes
Send To Partner	(Read only; value depends on action type)	**true** indicates an *out*bound (Send ToPartner) action. **false** indicates an *in*bound (Receive FromPartner) action.
Delivery Channel	(dropdown)	Choose from a dropdown list of XDC bindings that have been defined for this TP.
Internal Delivery Channel	[None]	If your host uses IDCs, choose from a dropdown list of IDC bindings that have been defined for this TP.

Table 11 Parameters in the (TP->TPP-> Business Service) "Business Actions" Tab (Continued)

Parameter Name	Default Value	Description / Notes
Enveloping Channel	(dropdown)	Choose from a dropdown list of XDC bindings that have been defined for this TP.
Send Acknowledgments	[None]	If set to **true**, then an error is issued if no CONTRL is received within the Time-Out period (see below). The value set here — either **true** or **false** — appears in Message Tracking of Request/Response.
Character Set Encoding		(Leave blank for default character set encoding.)
Send warnings with ACKs	[None]	
Batch Tracking	Both	Retain the default setting to assure tracking of outbound and inbound batches.
Group Tracking	Both	Retain the default setting to assure tracking of outbound and inbound groups.
Transaction Tracking	Both	Retain the default setting to assure tracking of outbound and inbound transactions.
Time-Out (Minutes)	10	Set the maximum amount of time to wait before issuing an error. Even if received afterward, Message Tracking still shows the timeout error occurred.
Business Protocol Validation Handler	[None]	Choose a handler BP of this type from the list. The list is populated by values in the B2B host.
Business Message Syntax Validation Handler	[None]	Required for all actions except CONTRL. Choose a handler BP of this type from the list.
Business Transaction Type Handler	[None]	Choose a handler BP of this type from the list. The list is populated by values in the B2B host.
Custom Business Protocol Validation Handler	[None]	If your B2B host has a handler BP of this type, choose it from the dropdown list.
Business ACK Generator Handler	[None]	If your B2B host has a handler BP of this type, choose it from the list.
Business ACK Processor Handler	[None]	Choose a handler BP of this type from the list.
Error Handler	[None]	If your B2B host uses error handler BPs, choose one from the list.
Custom External Unique ID Handler	[None]	If your B2B host has a custom handler BP of this type, choose it from the list.

Chapter 6

Working with the UN/EDIFACT Sample

The UN/EDIFACT Manager Composite Application comes with a sample ICAN Project. You can import this Project into Enterprise Designer and use it to quickly learn how to set up UN/EDIFACT Projects and business logic.

What's in This Chapter

6.1 About the UN/EDIFACT Manager Sample

The UN/EDIFACT Manager sample includes several Projects that you can import into Enterprise Designer to see how ICAN Projects for UN/EDIFACT are designed.

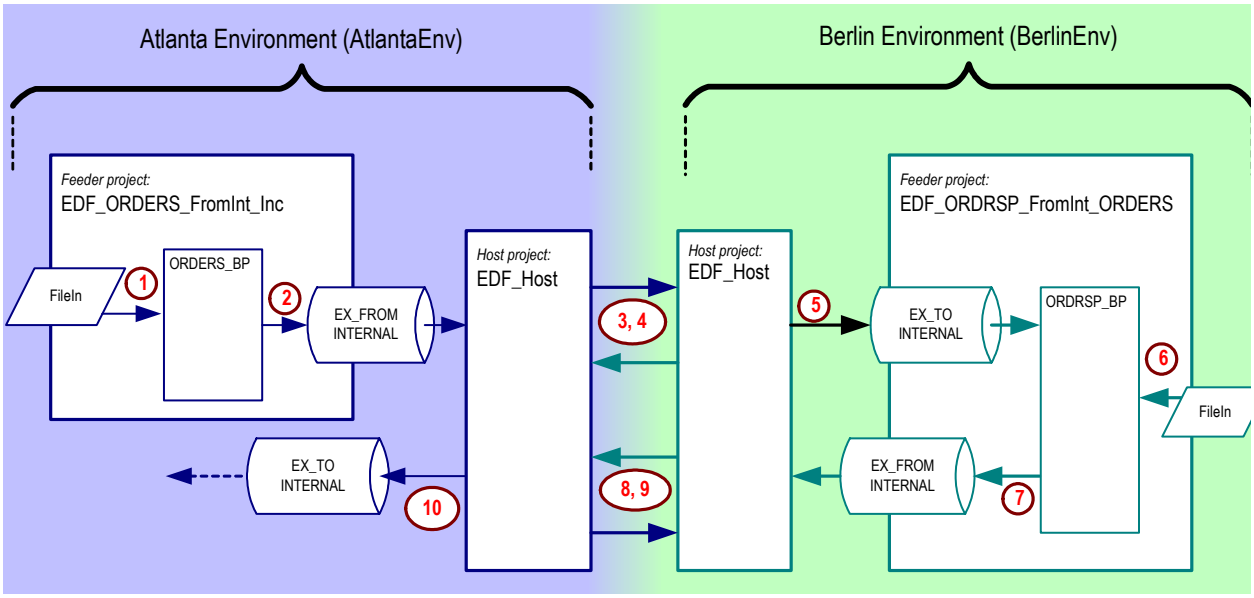
The sample demonstrates how UN/EDIFACT Manager is used for outbound and inbound message processing, exchanging EDIFACT ORDERS requests and ORDRSPresponses with CONTRL acknowledgments between enterprises named "Atlanta" and "Berlin".

The following section describes the sample process flow in detail.

6.1.1 Process Flow in the UN/EDIFACT Sample

The following figure shows the process flow and most important components in the sample. The numbered items in the following diagram shows the message flow.

Figure 4 UN/EDIFACT Sample Process Flow



Process Flow in the Atlanta Environment

Initiate processing and prepare/pass the ORDERS action

1. The Atlanta feeder Project, **EDF_ORDERS_FromInt_Inc**, is triggered by an inbound **File** eWay when it detects a file of the form **EDF_dlg_ORDERS_Out*.in**. It passes XML data to the **ORDERS_BP** process; the data identifies a trading partner and specifies a service, an action, and a source of data. (For the presupplied **.in** file, the TP is named **Berlin**, the service is **dlg_ORDERS_Out**, the action is **ORDERS**, and the data comes from a well-qualified filename.)
2. Using these four data items, the **ORDERS_BP** process obtains and modifies the ExStdEvent message and publishes to a standard eXchange topic named **EX_FROMINTERNAL**.

   One of this topic's subscribers is a business process (bpEX_MainFromTP) that provides ExStdEvent messages to the eXchange Service for the B2B host for Atlanta **EDF_Host**.

   - For more information about ExStdEvent, standard eXchange JMS topics lke EX_FROMINTERNAL, and core BPs like bpEX_MainFromTP, see the *eXchange Integrator Protocol Designer's Guide*.
   - For more information about the **ORDERS_BP**, see **"About the ORDERS Project BP" on page 47**.
3. Based on the specified service and action (**dlg_ORDERS_Out** and **ORDERS**) in the ExStdEvent message, the UN/EDIFACT Manager and eXchange process the message, and then the Atlanta host passes an ORDRSP response to the specified trading partner.

Process Flow in the Berlin Environment

Validate the ORDERS request, acknowledge, and prepare/pass the ORDRSP

1. When Berlin's B2B host (also named **EDF_Host**) receives the inbound ORDERS request from its trading partner Atlanta, it validates the message (per a business message syntax validation selector/handler) and sends back a CONTRL acknowledgment.
2. One of **EDF_Host**'s BPs (bp_Ex_BatchSplitter_EDIFACT) publishes the ExStdEvent message to a standard eXchange topic named **EX_TOINTERNAL**, one of whose subscribers is a sample process named **ORDRSP_BP**.
3. The **ORDRSP_BP** process, having obtained the ExStdEvent message, uses the Batch eWay to read information to generate a response. For more information on **ORDRSP_BP**, see **"About the ORDRSP Project BP" on page 50**.
4. The **ORDRSP_BP** process creates a new ExStdEvent message containing the ORDRSP response and publishes it to the standard eXchange topic **EX_FROMINTERNAL**. As before, a subscriber (bpEX_MainFromInternal) in the eXchange Service provides the message to the B2B host, **EDF_Host**.
5. Based on the specified service and action (**dlg_ORDERS_In** and **ORDRSP**) that it finds in the ExStdEvent, the Berlin host passes an ORDRSP response to the specified trading partner (**Atlanta**).

On the Atlanta side: Validate the ORDRSP response, acknowledge, and continue

6. When Atlanta's B2B host (**EDF_Host**) receives the inbound ORDRSP request from its trading partner Berlin, it validates the message (per a business message syntax validation handler) and sends back a CONTRL acknowledgment.

One of the **EDF_Host**'s BPs (bp_Ex_BatchSplitter_EDIFACT) publishes the message to the standard eXchange topic **EX_TOINTERNAL**.

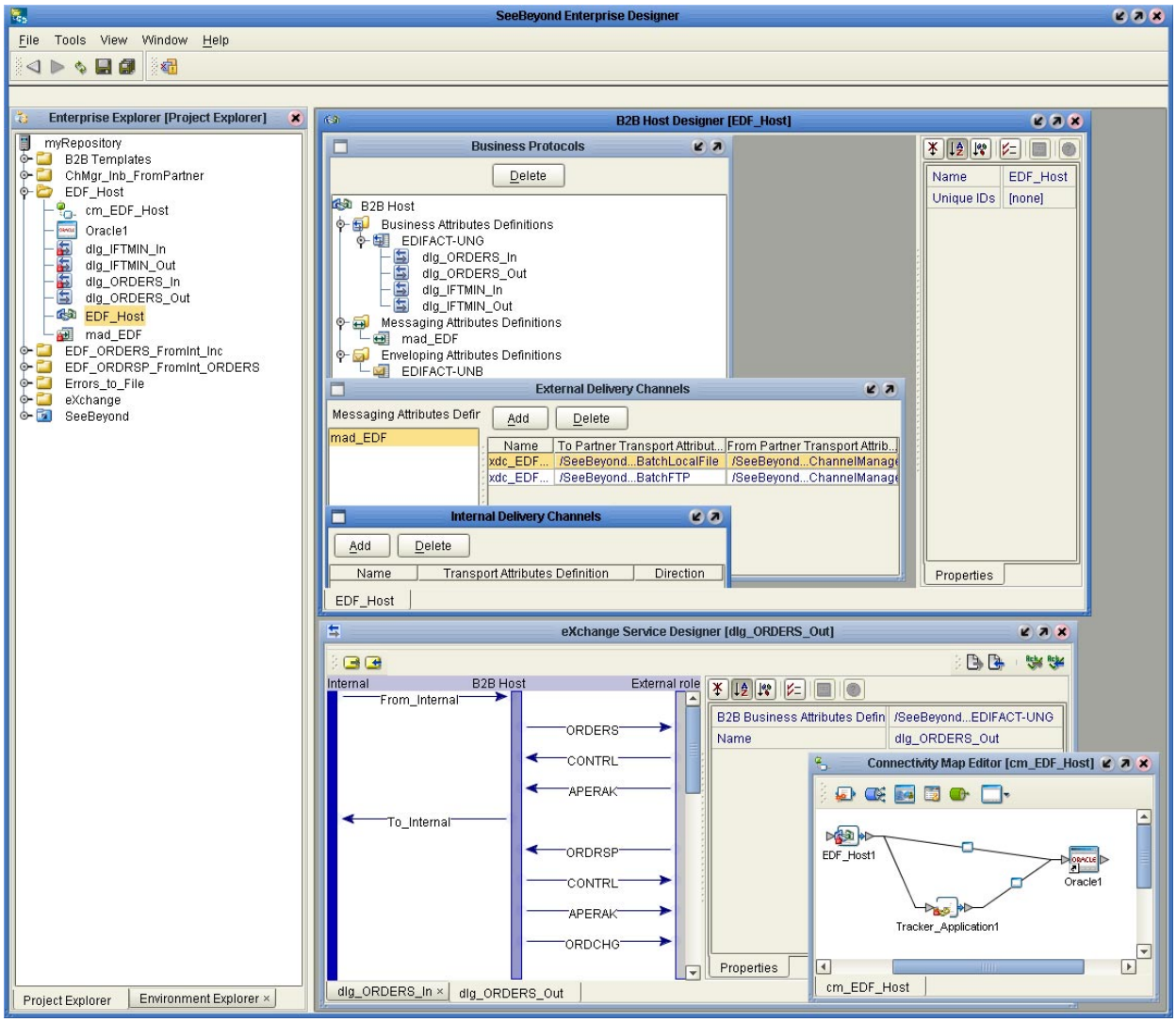
6.1.2 About the EDF_Host

- **dlg_<*action*>_Out** and **dlg_<*action*>_In** are the host project's services, also called "business dialogs":
  - The **dlg_<*action*>_Out** dialog has six actions (see **Figure 5 on page 42**): It starts with an outbound internal-to-host action, then continues with an outbound-to-TP action and an acknowledgment, followed by an inbound-from-TP reply and an acknowledgment; and it ends with an inbound host-to-internal action.
  - The **dlg_<*action*>_In** dialog also has six actions: It starts with an inbound-from-TP action and an acknowledgment, then continues with an inbound host-to-internal action, followed by an outbound internal-to-host action; and it ends an outbound-to-TP reply action and an acknowledgment.
- **mad_EDF** is the host project's message attributes definition (MAD).
- **EDF_Host** is the name of the B2B host itself, as well as the project name:

- Its Business Protocols window references the services (business dialogs) named **dlg_<*action*>_Out** and **dlg_<*action*>_In**. To find them, open the BADs folder and open the BAD named "**EDIFACT**".
- This window also references the MAD, **mad_EDF**. To find it, open the MAD folder.
- Its External Delivery Channels window defines two delivery channels for the UN/EDIFACT MAD. The sample implementation uses channels named **BLF_***, which reference the BatchLocalFile transport attributes definition (TAD) as their *sender* (*to* partner) transport protocol, and a ChannelManagerFile TAD as their *receiver* (*from* partner) transport protocol.

- **cm_EDF_Host** is the map of which activation creates the "eXchange Service" external:
  - Its only input is an instance of **EDF_Host**, with two connections going out (towards the right).
  - Its only output is an instance of Oracle, with two connections coming in (from the left).
  - Connecting to both is an instance of a SeeBeyond-supplied tracking application.

Figure 5 B2B Host (**EDF_Host**)— Project Components



6.2 Quick Steps to Get the Sample Up and Running

The following provides an overview of the steps necessary to get the sample up and running as quickly as possible. For detailed information, refer to the cross reference mentioned for each step.

To configure and run the UN/EDIFACT Manager sample:

1. Unzip the UN/EDIFACT Manager sample file in the following folder as described in **"Unzipping the Sample File" on page 44**.

   C:\temp\eXchange

2. Import the sample Project **.zip** file from the following folders as described in **"Importing the Sample Projects" on page 45**.

C:\temp\eXchange\Sample\EDIFACT\Projects
C:\temp\eXchange\Sample\Common\Projects

3 Configure the server name for the Oracle external application **myExtOracleOut** as described in **"Configuring the Oracle External Application" on page 52**.

4 For each Project, create, automap, and activate Deployment Profiles for each Project as defined in the table below. The Extra Steps column define additional steps you must take before activating profiles.

For details, refer to **"Creating and Activating Deployment Profiles" on page 53**.

Table 12 Deployment Profiles To Be Created and Activated

Project Name	Deployment Profile(s)	Extra Steps
EDF_Host	2 profiles; one for AtlantaEnv and a second for BerlinEnv	--
ChMgr_Inb_FromPartner	2 profiles; one for AtlantaEnv and a second for BerlinEnv	Assign an eXchangeService connection to EDF_Host eXchange Service for each profile.
Errors_to_File	2 profiles; one for AtlantaEnv and a second for BerlinEnv	Map to the **Directory** constant in each Environment before activating profiles.
EDF_ORDERS_FromInt_Inc	1 profile for AtlantaEnv	Map to the **DataDir** constant in the Environment before activating the profile.
EDF_ORDRSP_FromInt_ORDERS	1 profile for BerlinEnv	Map to the **DataDir** constant in the Environment before activating the profile.

If the error "Failed to obtain JNDI name prefix(ERROR_GET_JNDINAMEPREFIX)" occurs, you need to map to the **DataDir** constant.

5 Import and activate the Trading Partners in the eXchange Partner Manager by following the steps below. For details, refer to **"Importing and Activating Trading Partners" on page 54**.

A In the Configuration tab, click **Import**.

B Open the Repository and **BerlinEnv** and click **EDF_Host1**.

C Enter **Atlanta** for the name, browse to **C:\temp\eXchange\Sample\TradingPartners\EDIFACT_TP_for-BerlinEnv_to-from_Atlanta.xml**, and click **Import**.

D Do the same to create the Berlin TP for the AtlantaEnv but name the Trading Partner Berlin and point it to **C:\temp\eXchange\Sample\TradingPartners\EDIFACT_TP_for-AtlantaEnv_to-from_Berlin.xml**.

6 In the ***LogicalHost*\bootstrap\bin** folder, start the **AtlantaEnv** and **BerlinEnv** Logical Hosts with the following syntax:

bootstrap -r http://*myBox:nnnnn/myRepository* -i *myId* -p *myPassword* -e *EnvironmentName* -l LogicalHost1

For details, refer to **"Running the EDIFACT Sample" on page 54**.

7 Rename the **.~in** extension to **.in** in the **c:\temp\eXchange\Sample\EDIFACT\Data\Atlanta** folder.

6.3 Unzipping the Sample File

The UN/EDIFACT Manager sample Projects are included in the **EDIFACT_ManagerDocs.sar**. This file is uploaded separately from the UN/EDIFACT Manager product file (**EDIFACT_Manager.sar**) during installation. For information about uploading the **EDIFACT_ManagerDocs.sar,** refer to **Installing the UN/EDIFACT Composite Application** on page 27.

Once you have uploaded the **EDIFACT_ManagerDocs.sar** to the Repository and you have downloaded the sample Project (**EDIFACT_Manager_Sample.zip**) using the **DOCUMENTATION** tab in the Enterprise Manager, the sample resides in the folder specified during the download, which should be **c:\temp\eXchange**.

Note: *If you unzip the sample Project* ***.zip*** *file to a different location, modifications will be required. We recommend that you unzip the sample Project file to* ***c:\temp\eXchange*** *to avoid complications. Modifications would entail changing the value of the data root directory Project variables in Enterprise Designer, the fields in the ToPartner and FromPartner tabs in eXchange Partner Manager (), and the contents of the sample data files named* ***EDIFACT_dlg_****.*

Unzip the **EDIFACT_Manager_Sample.zip** file in the **c:\temp\eXchange** folder. The folder now contains the data files and importable Projects that make up the UN/EDIFACT sample.

The Projects are in the following folders:

- **c:\temp\eXchange\Sample\Common\Projects** (eXchange Project and Environment)
- **c:\temp\eXchange\Sample\EDIFACT\Projects** (UN/EDIFACT Projects)

The table below describes the purpose of each Project **.zip** file:

Table 13 EDIFACT_Manager_Sample.zip Project Files

Project File Name	Description
eX_Common_Projects.zip	eXchange Project
eX_Common_Environments.zip	eXchange Environments
EDF_ORDERS_ORDRSP_feeders.zip	Input Project
EDF_Host.zip	UN/EDIFACT Logical Host Project

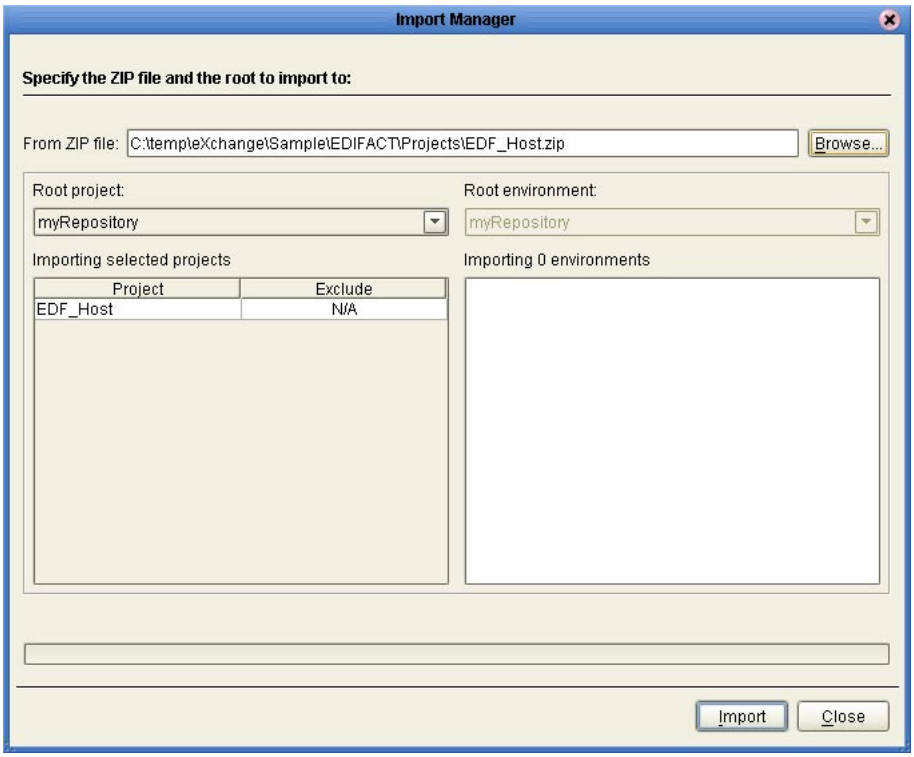
6.4 Importing the Sample Projects

After unzipping the sample **.zip** file as described in the previous section, you can import the UN/EDIFACT Manager sample into Enterprise Designer. The procedure below describes how you import the Project **.zip** files included in the UN/EDIFACT sample.

To import the sample Projects

1. In the **Project Explorer** tab of the Enterprise Designer, right-click the Repository and click **Import**. A message confirms if you want to save your changes.
2. Click **Yes** to save your changes. The **Import Manager** dialog box appears.
3. Click **Browse**, navigate to the **C:\temp\eXchange\Common\Projects** folder, and click **eX_Common_Projects.zip**.
4. Click **Open**. The **Import Manager** dialog box appears.

Figure 6 Importing Sample Projects



5. Click **Import**. If an "Missing APIs" warning appears, click **Continue**.

   If another error message occurs, a *.sar file has not been uploaded. Verify that all ***.sar** files required for the sample have been installed. For information, refer to **"Installing the UN/EDIFACT Composite Application" on page 27**.
6. Click **OK** at the dialog box confirming that the Project imported successfully.
7. Click **Browse** and repeat steps 5 and 6 to import **eX_Common_Environments.zip**.

8 Click **Browse** and navigate to the **c:\temp\eXchange\EDIFACT\Projects** folder.

9 Repeat step 5 and 6 to import the following Projects:

- **EDF_ORDERS_ORDRSP_feeders.zip**
- **EDF_Host.zip**

10 Click **Close**.

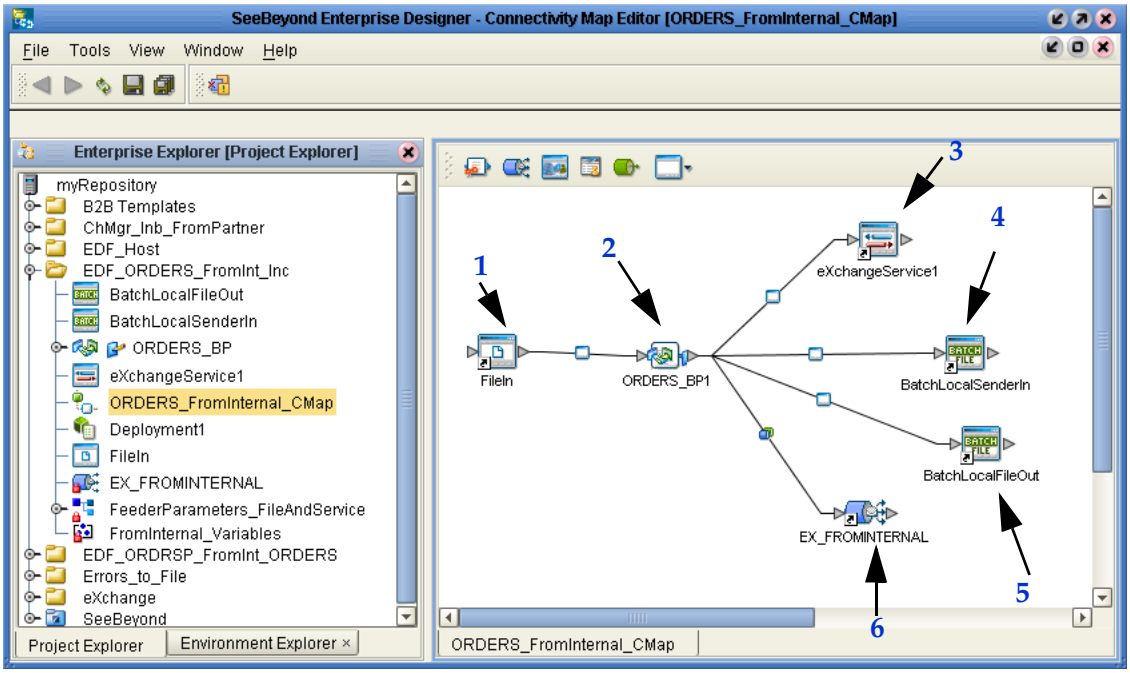
6.5 Understanding the ORDERS Feeder Project

This section describes the Connectivity Map and BPs for the ORDERS feeder Project.

6.5.1 About the ORDERS Project Connectivity Map

The request feeder Project, **EDF_ORDERS_FromInt_Inc**, has a Connectivity Map named **ORDERS_FromInternal_CMap**. This is illustrated below and explained in the callouts that follow Figure 7.

Figure 7 Connectivity Map for ORDERS Feeder Project



1 ***FileIn** eWay:* Connects to **ORDERS_BP1** and supplies the TP, service, action, and direction information required to look up a Trading Partner, as well as a data source for the initial ExStdEvent payload.

2 ***ORDERS_BP1** Business Process:* Receives the data, looks up the trading partner, gets ExStdEvent, increments the ID, updates ExStdEvent, and publishes it to the

EX_FROMINTERNAL topic. For more information, refer to **"About the ORDERS Project BP" on page 47**.

3 ***eXchangeService1****:* Provides services for the Atlanta host used by **ORDERS_BP1** for retrieving the ExStdEvent message and looking up the Trading Partner.

4 ***BatchLocalFileOut*** *eWay:* Writes out the Orders file for Berlin.

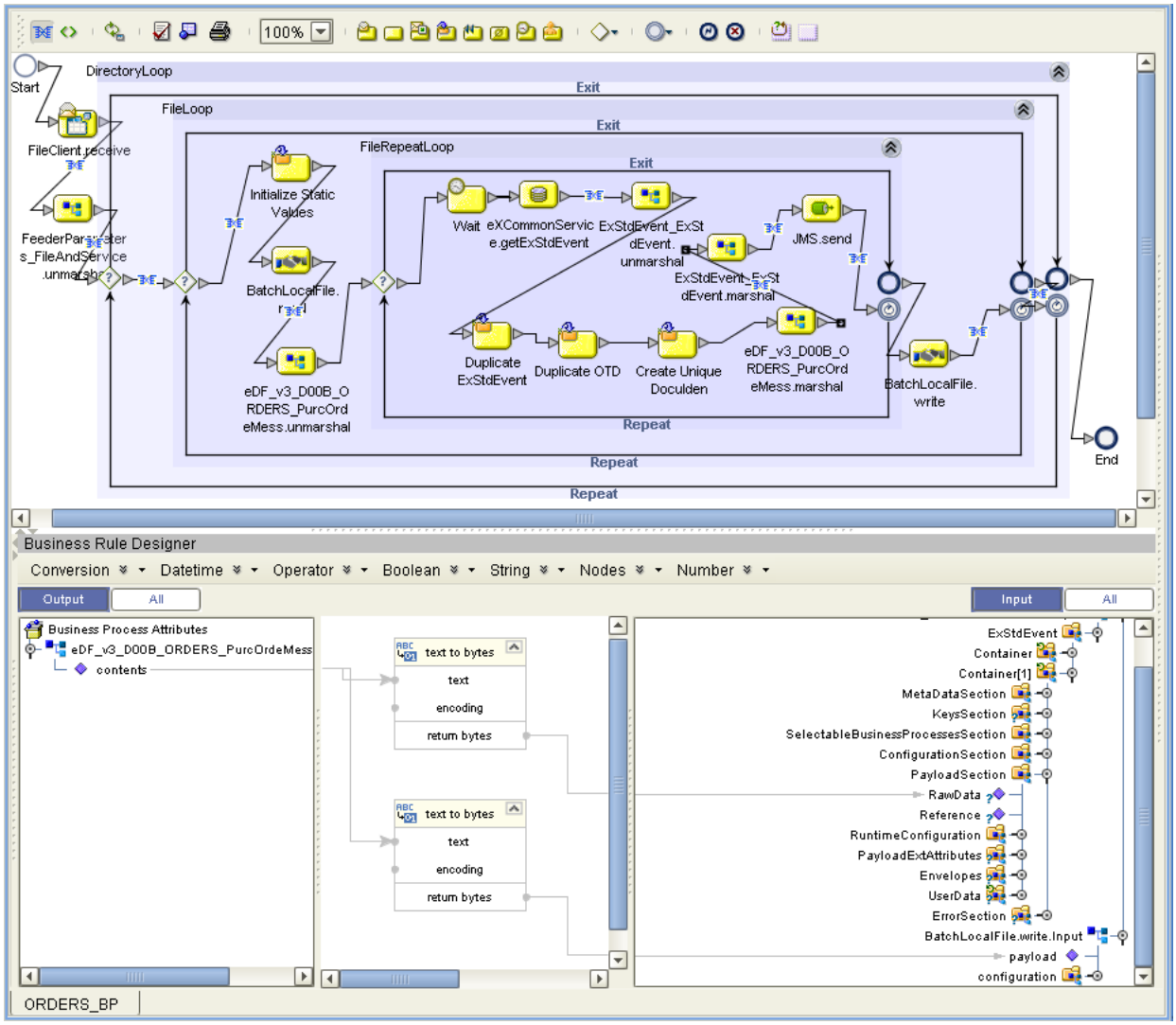
5 ***BatchLocalSenderIn*** *eWay:* Reads the Orders template file.

6 ***EX_FROMINTERNAL*** *topic:* A JMS topic whose publisher is ICAN and whose subscriber is the **bpEX_MainFromInternal** BP, which provides the message to the eXchange Service. For more information on standard eXchange JMS topics, see the *eXchange Integrator Protocol Designer's Guide*.

6.5.2 About the ORDERS Project BP

The **ORDERS_BP1** initiates the data flow for the ORDERS feeder Project. Figure 8 shows the **ORDERS_BP1** in the Enterprise Designer.

Figure 8 ORDERS_BP1 for the ORDERS Feeder Project



ORDERS_BP1: Initiating the data flow

1. ***receive** Activity* (of the "FileClient" service, from **SeeBeyond** > **eWays** > **File**): Receives the message.
2. ***unmarshal** Activity* (of the "FeederParameters_[...]" OTD, in the Project): Unmarshals the message.
3. ***Assign FeederParams** business rule:* Assigns the file information parameters, such as filename and location set in the Trading Partner Profile.
4. ***read** Activity* (of the "BatchLocalFile" service, from **SeeBeyond** > **eWays** > **BatcheWay**): Reads the input file.
5. ***Wait** timer:* Waits with processing and calling the standard event.
6. ***getExStdEvent** Activity* (of "eXCommonService", from **SeeBeyond** > **eXchange** > **Core Services**): Retrieves the ExStdEvent to prefill the next standard event.

7 ***unmarshal*** *Activity* (of the **ExStdEvent** OTD, from **SeeBeyond** > **eXchange** > **Core Components**): Unmarshals the input message into the standard event.

8 ***Duplicate ExStdEvent*** *business rule:* Duplicates the standard event.

9 ***unmarshal*** *Activity* (of the ORDERS_PurcOrde" OTD, from **SeeBeyond** > **OTD Library** > **EDIFACT** > **v3**): Unmarshals the template data that was read in.

10 ***Increment PurcOrdeNumb*** *business rule:* Increments the purchase order number.

11 ***marshal*** *Activity* (of the ORDERS_PurcOrde" OTD, from **SeeBeyond** > **OTD Library** > **EDIFACT** > **v3**): Creates the purchase order.

12 ***marshal*** *Activity* (of the "ExStdEvent" OTD, from **SeeBeyond** > **eXchange** > **Core Components**): Marshals the standard event.

13 ***send*** *Activity* (of the "JMS" OTD, from **SeeBeyond** > **eGate**): Sends the message to the JMS IQ Manager.

14 ***Increment FileRepCntr*** *business rule:* Increments the template file number.

15 ***write*** *Activity* (of the "BatchLocalFile" service, from **SeeBeyond** > **eWays** > BatcheWay): Writes the ORDERS out for the Trading Partner to retrieve.

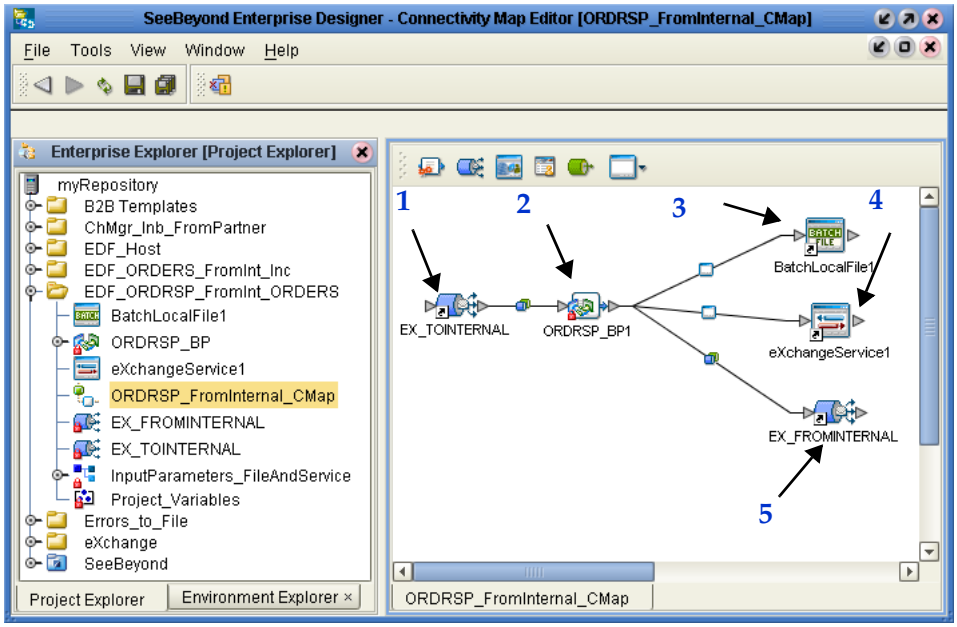
6.6 Understanding the ORDRSP Feeder Project

This section describes the Connectivity Map and BPs for the ORDRSP feeder Project.

6.6.1 About the ORDRSP Project Connectivity Map

The response feeder Project, EDF_ORDRSP_FromInt_ORDERS, has a Connectivity Map named **ORDRSP_FromInternal_CMap**. This is illustrated below and explained in the callouts that follow Figure 9.

Figure 9 Connectivity Map for ORDRSP Feeder Project



1 ***EX_TOINTERNAL** topic:* A JMS topic whose publisher is the **bpEX_SendToInternalAndDialogEDF BP**, which provides the message from the trading partner, and whose subscriber is ICAN.

2 ***ORDRSP_BP1**:* Receives the data, looks up the Trading Partner, retrieves **ExStdEvent**, increments the ID, updates **ExStdEvent**, and publishes it to the **EX_FROMINTERNAL** topic. For more information, refer to **"About the ORDRSP Project BP" on page 50**.

3 ***BatchLocalFile** eWay:* Writes out the ORDRSP template file.

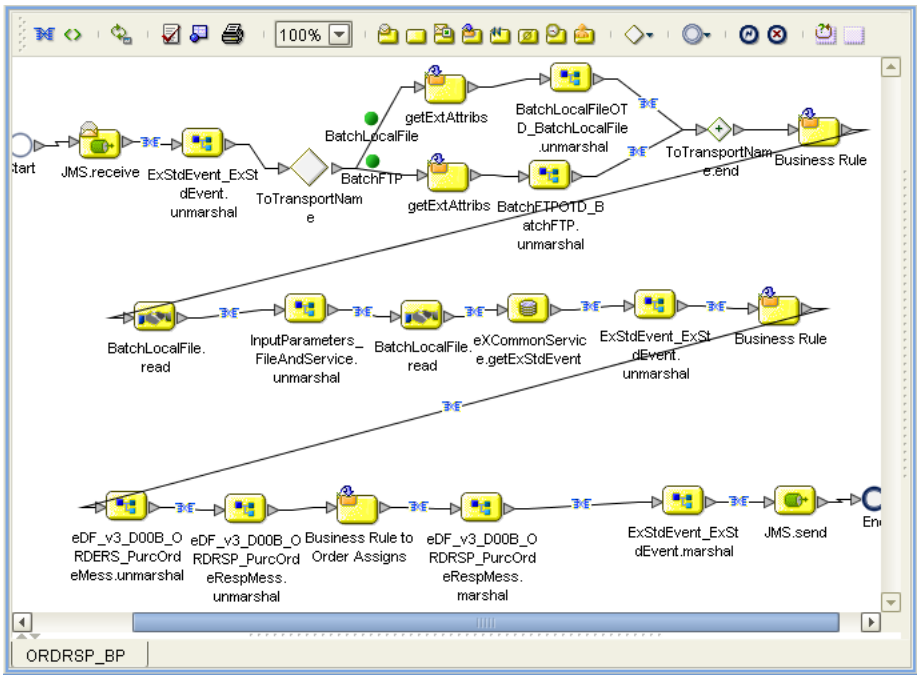
4 ***eXchangeService1**:* Provides services for the Berlin B2B host used by **ORDRSP_BP1** for retrieving the standard event and looking up the Trading Partner.

5 ***EX_FROMINTERNAL** topic:* A JMS topic whose publisher is ICAN and whose subscriber is the **bpEX_MainFromInternal** BP which provides the message to the eXchange Service.

6.6.2 About the ORDRSP Project BP

The **ORDRSP_BP1** replies to a request for the ORDERS feeder Project. Figure 10 shows the **ORDRSP_BP1** in the Enterprise Designer.

Figure 10 ORDRSP_BP1 for the ORDRSP Feeder Project



1 ***receive** Activity* (of the JMS OTD, from **SeeBeyond** > **eGate**): Receives message.

2 ***unmarshal** Activity* (of the **ExStdEvent** OTD, from **SeeBeyond** > **eXchange** > **Core Components**): Unmarshals the message using the standard event OTD.

3 ***Delivery Attributes** business rules:* Assigns delivery attributes (file, location) as defined in the Trading Partner Profile.

4 ***unmarshal** Activity* (of the "BatchFTP" or "BatchLocalFile" service, from **SeeBeyond** > **eWays** > **BatcheWay**): Sets all batch parameters to indicate to the read Activity which files to pick up and where.

5 ***Set Response Service Param FileName** business rule:* Dynamically creates the file to be retrieved.

6 ***read** Activity* (of the "BatchLocalFile" service, from **SeeBeyond** > **eWays** > **BatcheWay**): Reads the ORDRSP message.

7 ***unmarshal** Activity* (of the "InputParameters_[...]" OTD, in the Project): Fills in the next read Activity with parameter information.

8 ***read** Activity* (of the "BatchLocalFile" service, from SeeBeyond > eWays > BatcheWay): Reads the ORDRSP template.

9 ***getExStdEvent** Activity* (of "eXCommonService", from **SeeBeyond** > **eXchange** > **Core Services**): Generates an updated standard event.

10 ***unmarshal** Activity* (of the **ExStdEvent** OTD, from **SeeBeyond** > **eXchange** > **Core Components**): Unmarshals the updated standard event.

11 ***Business Rule***(empty): A marker indicating the last activity in the business process that must not be modified.

12 ***unmarshal** Activity* (of the **ORDERS_PurcOrde** OTD, from **SeeBeyond** > **OTD Library** > **EDIFACT** > **v3**): Unmarshals the ORDERS message.

13 ***unmarshal** Activity* (of the **ORDRSP_PurcOrde** OTD, from **SeeBeyond** > **OTD Library** > **EDIFACT** > **v3**): Generates the ORDRSP message.

14 ***Duplicate Response** business rule:* Verifies whether there are duplicates.

15 ***marshal** Activity* (of the **ORDRSP_PurcOrde** OTD, from **SeeBeyond** > **OTD Library** > **EDIFACT** > **v3**): Marshals the ORDRSP message.

16 ***marshal** Activity* (of the **ExStdEvent** OTD, from **SeeBeyond** > **eXchange** > **Core Components**): Marshals the standard event.

17 ***send** Activity* (of the JMS OTD, from **SeeBeyond** > **eGate**): Sends the message to the JMS IQ Manager.

6.7 Configuring the Oracle External Application

The procedure below describes how to configure the Oracle external application for the sample. As described in **"Configuring the Oracle Database" on page 29**, the database must have the users ex_A and ex_B. The only other configuration to change is the server name. By default, the server name is localhost.

To configuring the Oracle External Application

1 In the **Environment Explorer** tab in the Enterprise Designer, expand **AtlantaEnv**, right-click **myExtOracleOut**, and click **Check Out**.

2 Right-click **myExtOracleOut** and click **Properties**.

3 Enter the server name for your Oracle database.

4 Verify that the other options are configured appropriately as follows:

Option	Setting
DatabaseName	*SID for the eXchange database*
DataSourceName	**local**
Password	**ex_A**
PortNumber	**1521** (unless the Oracle administrator changed the default)
ServerName	*name of the Oracle server*
User	**ex_A**

5 Click **OK** to close the **Properties** dialog box.

1 Expand **BerlinEnv**, right-click **myExtOracleOut**, and click **Check Out**.

2 Right-click **myExtOracleOut** and click **Properties**.

3 Enter the server name for your Oracle database and verify that the other options are correct as described in step 4. The user name and password must be **ex_B**.

6.8 Creating and Activating Deployment Profiles

After creating the validation Connectivity Map, create, automap, and activate Deployment Profiles for each Project as defined in the table below. The Extra Steps column defines additional steps you must take before activating profiles.

Table 14 Deployment Profiles To Be Created and Activated

Project Name	Deployment Profile(s)	Extra Steps
EDF_Host	2 profiles; one for AtlantaEnv and a second for BerlinEnv	
ChMgr_Inb_FromPartner	2 profiles; one for AtlantaEnv and a second for BerlinEnv	Assign an eXchangeService connection to EDF_Host eXchange Service for each profile.
Errors_to_File	2 profiles; one for AtlantaEnv and a second for BerlinEnv	Map to the **Directory** constant in each Environment before activating profiles.
EDF_ORDERS_FromInt_Inc	1 profile for AtlantaEnv	Map to the **DataDir** constant in the Environment before activating the profile.
EDF_ORDRSP_FromInt_ORDERS	1 profile for BerlinEnv	Map to the **DataDir** constant in the Environment before activating the profile.

Note: *If the error "Failed to obtain JNDI name prefix(ERROR_GET_JNDINAMEPREFIX)" occurs, you forgot to map to the* **DataDir** *constant.*

The procedure below describes how to create, automap, and activate the Deployment Profiles listed in the table above.

To create and activate Deployment Profiles

1. In the **Project Explorer** tab of Enterprise Designer, right-click the Project for which you are adding a Deployment Profile, click **New**, **Deployment Profile**, enter a name, click **AtlantaEnv** or **BerlinEnv** and click **OK**.
2. For the following Projects, click **Map Variables**, click **Mapped Name**, click **DataDir**, and click **OK**.
   - **Errors_to_File**
   - **EDF_ORDERS_FromInt_Inc**
   - **EDF_ORDRSP_FromInt_ORDERS**
3. Click **Automap** and click **OK** when the confirmation message appears.

   Both **ChMgr_Inb_FromPartner** profiles require you to assign an eXchangeService connection to the EDF_Host eXchange Service; select protocol **mad_EDF**.
4. Click **Save All**.
5. Click **Activate**. Click **No** when the Apply to Logical Host message appears.

6.9 Importing and Activating Trading Partners

After activating the Deployment Profiles as described in the previous section, import and activate Trading Partners in ePM as described in the procedure below.

You are importing a Trading Partner configured for either the Atlanta or the Berlin Environment. In **BerlinEnv**, you take the viewpoint of the Berlin host: "ToPartner" means "to Atlanta"; "FromPartner" means "from Atlanta". In **AtlantaEnv**, you take the viewpoint of the Atlanta host: "ToPartner" means "to Berlin"; "FromPartner" means "from Berlin".

The Repository and Oracle database must be running and accessible for this procedure.

For detailed information about setting up Trading Partners in ePM, refer to the *eXchange Integrator User's Guide*.

To import and activate the Trading Partners

1. In the **Configuration** tab in ePM, click **Import**.
2. Open the Repository and **BerlinEnv** and click **EDF_Host1**.
3. Enter **Atlanta** for the name, browse to:

   C:\temp\eXchange\Sample\TradingPartners\EDIFACT_TP_for-BerlinEnv_to-from_Atlanta.xml, and click **Import**.
4. Do the same to create the Berlin TP for the AtlantaEnv but name the Trading Partner Berlin and point it to:

   C:\temp\eXchange\Sample\TradingPartners\EDIFACT_TP_for-AtlantaEnv_to-from_Berlin.xml.
5. Select Atlanta or Berlin, click **Activate**.
6. Click **Activate**.

6.10 Running the EDIFACT Sample

Once you have completed all the steps in the sections in this chapter, you are ready to run the sample. The section below describes how to start the Logical Hosts. After the Logical Hosts are running, change the file extensions of the input files in the data directory as described in **"Preparing the Input Data" on page 55**.

6.10.1 Starting the Logical Hosts

The procedure below describes how to start the Logical Hosts for the AtlantaEnv and BerlinEnv Environments.

To start the Logical Hosts

1. Navigate to the ***LogicalHost*\bootstrap\bin** directory, where *LogicalHost* is the Logical Host name for the **AtlantaEnv** or **BerlinEnv** Environment.

2 Use the following syntax to start the Logical Host

```
bootstrap -r http://myBox:nnnnn/myRepository -i myId -p myPassword
-e EnvironmentName -l LogicalHost1
```

Where *myBox* is the host name, *nnnnn* is the host port number, *myRepository* is the Repository name, *myID* is the ID, *mypassword* is the password, and *EnvironmentName* is **AtlantaEnv** or **BerlinEnv** depending on what you entered in step 1.

3 Repeat steps 1 and 2 to start the second Logical Host.

6.10.2 Preparing the Input Data

Once both Logical Hosts are up and running, go to the data directories and rename the input files so that they are picked up by the sample.

To prepare the input data

- In the **c:\temp\eXchange\Sample**\EDIFACT**Data\Atlanta** folder, change the file extension **.~in** to **.in** for the following file:

  EDF**_dlg_ORDERS_Out_feeder_Berlin.~in**

The Atlanta host ...

- Reads files of this form:
  ...\Sample\EDIFACT\Data\Atlanta**EDF_dlg_<*action*>_Out_feeder_Berlin.in**

  The **.in** file is an XML file that points to the **C:\temp\eXchange\Sample\Data\EDIFACT\Atlanta** folder and **_incremented.unh** file.
- Writes output resulting from successful processing to files of this form:
  ...\Sample\EDIFACT\Data**Berlin**\Atlanta_ORDERS_In_EDF_<*timestamp*>.dat
- Writes error messages to files of this form:
  ...\Sample\EDIFACT\Data\Atlanta**ProcessedError_%d.dat**
  ...\Sample\EDIFACT\Data\Atlanta**DeadLetter_%d.dat**

The Berlin host ...

- Reads files of this form:
  ...\Sample\EDIFACT\Data\Berlin\EDIFACT_*
- Writes output resulting from successful processing to files of this form:
  ...\Sample\EDIFACT\Data**Atlanta**\EDF_dlg_ORDERS_Out_EDF_<*timestamp*>.dat
- Writes error messages to files of this form:
  ...\Sample\EDIFACT\Data\Berlin**ProcessedError_%d.dat**
  ...\Sample\EDIFACT\Data\Berlin**DeadLetter_%d.dat**

Appendix A

OTD Syntax Validation BPs

This appendix provides background and conceptual information on OTD syntax validation BPs.

What's in This Appendix

A.1 Activity Flow

The activity flow of an OTD syntax validation handler B2B protocol process consists of these eight steps: (1) Receive input; (2) Copy ExStdEvent; (3) Concatenate the payload's headers/data/trailer into a string; (4) Unmarshal the concatenated string; (5) Populate BizAckCorrKey; (6) Set BizRespCorrKey; (7) Perform validate; (8) Check results.

These steps are discussed in detail below.

1. The BP receives the inbound message data from its invoker.
2. The BP copies ExStdEvent from inbound to outbound: In other words, copying the entire content of the **ExStdEvent** portion of the inbound data into the **ExStdEvent** portion of the outbound data for the handler.
3. The BP concatenates the following headers, data, and trailers, in order, from the **PayloadSection** part of the inbound data's **ExStdEvent**, and then copying this concatenated string into the contents part of input of **unmarshal**:
   A. Envelopes/BusinessProtocol/Batch/**Header**
   B. Envelopes/BusinessProtocol/Group/**Header**
   C. Envelopes/BusinessProtocol/Group/**Header**
   D. **RawData**
   E. Envelopes/BusinessProtocol/Group/**Trailer**
   F. Envelopes/BusinessProtocol/Batch/**Trailer**
4. The BP unmarshals the input string constructed in step 3.

If the unmarshaling process throws an UnmarshalException or GenericException, the exceptions are handled by fault handlers. See **"Fault Handling" on page 59**.

5 The BP populates **BizAckCorrKey** in the following four sub-steps:

- For the ExStdEvent part of the Handler's outbound data: Populate the KeysSection/CorrelationKeys/**BizAckCorrelationKey** with a string formed by concatenating by the following components, in order:

A PayloadSection/KeysSection/InternalIDs/**ExTradingPartnerGUID**
(this comes from the ExStdEvent part of inbound data).

B |
(the pipe character)

C PayloadSection/MetaDataSection/Event/**BusinessProtocolName**
(this comes from the ExStdEvent part of inbound data)

D |
(the "pipe" character)

E ${OuterNodeName}/${InnerNodeName}[1]/${TransNodeName}[1]/
${TSHeaderNodeName}/${BusinessTransactionIdentifier}
(this comes from the unmarshaled OTD).

F |
(the "pipe" character)

G ${OuterNodeName}/${InnerNodeName}[1]/${FGHeaderNodeName}/
${FGCtrlNumNodeName}
(this comes from the unmarshaled OTD)

H |
(the "pipe" character)

I ${OuterNodeName}/${InnerNodeName}[1]/${TransNodeName}[1]/
${TSHeaderNodeName}/${TransactionExternalID}
(this comes from the unmarshaled OTD)

- Assign ${OuterNodeName}/${InnerNodeName}[1]/${TransNodeName}[1]/ ${TSHeaderNodeName}/**${BusinessTransactionIdentifier}** (from the unmarshaled OTD) to MetaDataSection/Event/**BusinessTransactionIdentifier** (of the ExStdEvent part of Handler's outbound data)
- Assign ${OuterNodeName}/${InnerNodeName}[1]/${TransNodeName}[1]/ ${TSHeaderNodeName}/**${TransactionExternalID}** (from the unmarshaled OTD) to KeysSection/ExternalIDs/**TransactionExternalID** (of the ExStdEvent part of Handler's outbound data)
- Assign unmarshaled OTD to the OTD part of the validate input.

6 At this point, are two possibilities for setting **BizRespCorrKey**, depending on whether the inbound data is a request or a response. This is determined by the MetaDataSection/Event/BusinessTransactionType (of the ExStdEvent part of inbound data).

- In the case of a *request* (that is, the BusinessTransationType is 'Request'), then the BP sets BizRespCorrKey using BizTxID by populating KeysSection/

CorrelationKeys/**BizResponseCorrelationKey** (of the ExStdEvent part of the Handler's outbound data) with a string formed by concatenating the following components, in order:

A KeysSection/InternalIDs/**ExTradingPartnerGUID**
(from the ExStdEvent part of inbound data).

B **|**
(the "pipe" character)

C MetaDataSection/Event/**BusinessProtocolName**
(from the ExStdEvent part of inbound data)

D **|**
(the "pipe" character)

E MetaDataSection/Event/BusinessTransactionIdentifier
(from the ExStdEvent part of inbound data)

F |
(the "pipe" character)

G ${OuterNodeName}/${InnerNodeName}[1]/${TransNodeName}[1]/**${BizRespCorrPath}**
(from the unmarshaled OTD)

- In the case of a *response* or *reply* (that is, the BusinessTransationType is ***not*** 'Request'), then the BP sets BizRespCorrKey using BizTxID by populating the KeysSection/CorrelationKeys/**BizResponseCorrelationKey** (of the ExStdEvent part of the Handler's outbound data) with a string formed by concatenating the following components, in order:

A KeysSection/InternalIDs/**ExTradingPartnerGUID**
(from the ExStdEvent part of inbound data)

B **|**
(the "pipe" character)

C MetaDataSection/Event/**BusinessProtocolName**
(from the ExStdEvent part of inbound data)

D **|**
(the "pipe" character)

E MetaDataSection/Event/ProtocolRespondToMessageID
(from the ExStdEvent part of inbound data)

F |
(the "pipe" character)

G ${OuterNodeName}/${InnerNodeName}[1]/${TransNodeName}[1]/**${BizRespCorrPath}**
(from the unmarshaled OTD)

7 The BP performs a **validate** operation.

8 The BP checks the output of the validate operation and acts based on its severity. There are three cases: error, warning, or no problem.

- In the case of an *error* (in other words, when the contents of the output includes the string <Severity>ERROR</Severity>): The BP copies the validate output's contents into the message part of the validate fault, and throws this populated fault. The validate fault is then handled by the fault handler. See **"ValidateException" on page 59**.
- In the case of an *warning only* (that is, the output does not contain the string <Severity>ERROR</Severity> but is found to contain the string <Severity>WARN</Severity>), the BP populates the following fields in the ExStdEvent part of Handler's output before the Handler returns the populated ExStdEvent part to its invoker:
  - It assigns contents (from the validate output) to ErrorSection/ExException/ExceptionDetails/OTDParsingErrors/**ParsingErrorsXML**
  - It assigns '${OtdName}' to ErrorSection/ExException/ExceptionDetails/OTDParsingErrors/**OTDIdentifier**
  - It assigns 'BusinessMessageSyntaxValidation results' to ErrorSection/ExException/ExceptionDetails/**ErrorDescription**
  - It assigns 'OtdErrors' to ErrorSection/ExException/ExceptionDetails/**ErrorCode**
  - It assigns WARN' to ErrorSection/ExException/ExceptionDetails/**ErrorSeverity**
- In the case of *no problem* (that is, the output does not contain either string <Severity>ERROR</Severity> or <Severity>WARN</Severity>): The BP returns to its invoker. Some fields of the ExStdEvent part have been populated already, as noted in previous steps.

A.2 Fault Handling

This section takes a closer look at the Fault Handling activity flow mentioned in steps 4 and 8 in the previous section.

In this section

A.2.1. ValidateException

If a ValidateException (validate fault) is thrown during the validate operation, then the following additional assignments are performed to populate certain fields in the ExStdEvent part of the handler's output before the handler returns the populated ExStdEvent part to its invoker:

- The message part of the validate fault is assigned to ErrorSection/ExException/ExceptionDetails/OTDParsingErrors/**ParsingErrorsXML**
- The variable **${OtdName}** is assigned to ErrorSection/ExException/ExceptionDetails/OTDParsingErrors/**OTDIdentifier**
- The string literal **'BusinessMessageSyntaxValidation results'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorDescription**
- The string literal **'OtdErrors'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorCode**
- The string literal **'ERROR'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorSeverity**

A.2.2. UnmarshalException

If an UnmarshalException is thrown during the unmarshal operation, then the following additional assignments are performed to populate certain fields in the ExStdEvent part of the handler's output before the handler returns the populated ExStdEvent part to its invoker:

- The message part of the unmarshal fault is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorDescription**
- The string literal **'com.stc.otd.runtime.UnmarshalException'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorCode**
- The string literal **'ERROR'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorSeverity**

A.2.3. GenericException

If a GenericException is thrown during the unmarshal operation, then the following additional assignments are performed to populate certain fields in the ExStdEvent part of the handler's output before the handler throws the populated output as a fault:

- The message part of the unmarshal fault is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorDescription**
- The string literal **'Unknown Exception from otd unmarshal'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorCode**
- The string literal 'ERROR' is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorSeverity**

After the assignments are made, the BP throws the populated output as a fault. It is expected that a GenericException fault will be handled by the handler's invoker.

A.2.4. Other Faults

If another fault is thrown but not caught as a ValidateException, UnmarshalException, or GenericException, then the following additional assignments are performed to

populate certain fields in the ExStdEvent part of the Handler's output before the Handler throws the populated output as a fault.

- The string literal **'Unknown Error(s) Occurred in the OTD message syntax validation handler'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorDescription**
- The string literal **'Unknown Errors'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorCode**
- The string literal **'ERROR'** is assigned to ErrorSection/ExException/ExceptionDetails/**ErrorSeverity**

A.3 Variables Referenced by OTD Validation BPs

The table below lists the variables that the validation BPs reference. The assignment column uses an 850 of the X12 version 4010 as an example.

Table 15 Variables Referenced by OTD Validation BPs

Variable Name	Assignment (using as an example)
${OtdName}	_PurcOrde_Full
${HandlerName}	_Full_SynValHandler
${OuterNodeName}	_PurcOrde_Outer
${InnerNodeName}	_PurcOrde_Inner
${TransNodeName}	_PurcOrde
${TSHeaderNodeName}	_1_TranSetHead
${BusinessTransactionIdentifier}	E143_1_TranSetIdenCode
${TransactionExternalID}	E329_2_TranSetContNumb
${FGHeaderNodeName}	_FuncGrouHead
${FGCtrlNumNodeName}	E28_6_GrouContNumb
${BizRespCorrPath}	BEG_2_BegiSegmForPurcOrde/E324_3_PurcOrdeNumb
${BizRespCorrPath}	(see below)

A.3.1. The Value of the ${BizRespCorrPath} Variable

The value of the ${BizRespCorrPath} variable is obtained by looking up mapping tables with entries of *key=value* where *key* is the OTD name and *value* is the lookup value.

If lookup does not retrieve a value from mapping tables, then a default value is supplied based on the protocol (ASC X12, HIPAA, UN/EDIFACT, or EANCOM), as described below.

Default Value for ${BizRespCorrPath} in X12 v4010

Mapping table entries:

- X12_4010_270_EligCoveOrBeneInqu_Full=BHT_2_BegiOfHierTran/**E127_3_RefeIden**
- X12_4010_271_EligCoveOrBeneInfo_Full=BHT_2_BegiOfHierTran/**E127_3_RefeIden**
- X12_4010_850_PurcOrde_Full=BEG_2_BegiSegmForPurcOrde/**E324_3_PurcOrdeNumb**
- X12_4010_855_PurcOrdeAckn_Full=BAK_2_BegiSegmForPurcOrdeAckn/**E324_3_PurcOrdeNumb**
- X12_4010_997_FuncAckn_Full=ST_1_TranSetHead/**E329_2_TranSetContNumb**

Default value: **ST_1_TranSetHead/E329_2_TranSetContNumb**

Default Value for ${BizRespCorrPath} in X12 v4030

Mapping table entries:

- X12_4030_270_EligCoveOrBeneInqu_Full=BHT_2_BegiOfHierTran/**E127_3_RefeIden**
- X12_4030_271_EligCoveOrBeneInfo_Full=BHT_2_BegiOfHierTran/**E127_3_RefeIden**
- X12_4010_850_PurcOrde_Full=BEG_2_BegiSegmForPurcOrde/**E324_3_PurcOrdeNumb**
- X12_4010_855_PurcOrdeAckn_Full=BAK_2_BegiSegmForPurcOrdeAckn/**E324_3_PurcOrdeNumb**
- X12_4030_997_FuncAckn_Full=ST_1_TranSetHead/**E329_2_TranSetContNumb**

Default value: **ST_1_TranSetHead/E329_2_TranSetContNumb**

Default Value for ${BizRespCorrPath} in X12 v4061

Mapping table entries:

- X12_4061_270_EligCoveOrBeneInqu_Full=BHT_2_BegiOfHierTran/**E127_3_RefeIden**
- X12_4061_271_EligCoveOrBeneInfo_Full=BHT_2_BegiOfHierTran/**E127_3_RefeIden**
- X12_4061_850_PurcOrde_Full=BEG_2_BegiSegmForPurcOrde/**E324_3_PurcOrdeNumb**
- X12_4061_855_PurcOrdeAckn_Full=BAK_2_BegiSegmForPurcOrdeAckn/**E324_3_PurcOrdeNumb**

- X12_4061_997_FuncAckn_Full=ST_1_TranSetHead/**E329_2_TranSetContNumb**

Default value: **ST_1_TranSetHead/E329_2_TranSetContNumb**

Default Value for ${BizRespCorrPath} in Other X12 Versions

Default value: **ST_1_TranSetHead/E329_2_TranSetContNumb**

Default Value for ${BizRespCorrPath} in HIPAA Addenda

Mapping table entries:

- X12_004010X092A1_00_hipaa_270_EligCoveOrBeneInqu_Full=BHT_msk1_2_BegiOfHierTran/E127_3_RefeIden
- X12_004010X092A1_00_hipaa_271_EligCoveOrBeneInfo_Full=BHT_msk1_2_BegiOfHierTran/E127_3_RefeIden

Default value: **ST_1_TranSetHead/E329_2_TranSetContNumb**

Default Value for ${BizRespCorrPath} in HIPAA Standard

Mapping table entries:

- X12_004010X092_00_hipaa_270_EligCoveOrBeneInqu_Full=BHT_msk1_2_BegiOfHierTran/E127_3_RefeIden
- X12_004010X092_00_hipaa_271_EligCoveOrBeneInfo_Full=BHT_msk1_2_BegiOfHierTran/E127_3_RefeIden

Default value: **ST_1_TranSetHead/E329_2_TranSetContNumb**

Default Value for ${BizRespCorrPath} in UN/EDIFACT v3 and v4

Default value: **BGM_2_BegiOfMess/C106_2_DocuIden/E1004_1_DocuIden**

Default Value for ${BizRespCorrPath} in EANCOM v3 and v4

Default value: **BGM_2_BegiOfMess/C106_2_DocuIden/E1004_1_DocuIden**

Index

U

X