

*SeeBeyond ICAN Suite*

# ASC X12 Manager Composite Application User's Guide

*Release 5.0.1*



The information contained in this document is subject to change and is updated periodically to reflect changes to the applicable software. Although every effort has been made to ensure the accuracy of this document, SeeBeyond Technology Corporation (SeeBeyond) assumes no responsibility for any errors that may appear herein. The software described in this document is furnished under a License Agreement and may be used or copied only in accordance with the terms of such License Agreement. Printing, copying, or reproducing this document in any fashion is prohibited except in accordance with the License Agreement. The contents of this document are designated as being confidential and proprietary; are considered to be trade secrets of SeeBeyond; and may be used only in accordance with the License Agreement, as protected and enforceable by law. SeeBeyond assumes no responsibility for the use or reliability of its software on platforms that are not supported by SeeBeyond.

SeeBeyond, e\*Gate, e\*Way, and e\*Xchange are the registered trademarks of SeeBeyond Technology Corporation in the United States and/or select foreign countries. The SeeBeyond logo, SeeBeyond Integrated Composite Application Network Suite, eGate, eWay, eInsight, eVision, eXchange, eView, eIndex, eTL, ePortal, eBAM, and e\*Insight are trademarks of SeeBeyond Technology Corporation. The absence of a trademark from this list does not constitute a waiver of SeeBeyond Technology Corporation's intellectual property rights concerning that trademark. This document may contain references to other company, brand, and product names. These company, brand, and product names are used herein for identification purposes only and may be the trademarks of their respective owners.

© 2005 SeeBeyond Technology Corporation. All Rights Reserved.

**This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence.**

Version 20050408172217.

# Contents

List of Figures	7
-----------------	---

List of Tables	8
----------------	---

---

## Chapter 1

<b>Introduction</b>	<b>9</b>
<b>About This Document</b>	<b>9</b>
What's in This Document	9
Intended Audience	10
Document Conventions	10
Screenshots	10
<b>Related Documents</b>	<b>10</b>
<b>References</b>	<b>11</b>
<b>SeeBeyond Web Site</b>	<b>11</b>
<b>SeeBeyond Documentation Feedback</b>	<b>11</b>

---

## Chapter 2

<b>Overview of the ASC X12 Manager</b>	<b>12</b>
<b>About the ASC X12 Manager Composite Application</b>	<b>12</b>
<b>How the ASC X12 Manager Works</b>	<b>14</b>
<b>About the ASC X12 Protocol</b>	<b>14</b>
What Is X12?	14
What Is a Message Structure?	15
<b>Components of an X12 Envelope</b>	<b>16</b>
Data Elements	16
Segments	16
Loops	17
Delimiters	17
<b>Structure of an X12 Envelope</b>	<b>18</b>
Transaction Set (ST/SE)	19
Functional Group (GS/GE)	20
Interchange Envelope (ISA/IEA)	21

Control Numbers	22
ISA13 (Interchange Control Number)	22
GS06 (Functional Group Control Number)	22
ST02 (Transaction Set Control Number)	22
<b>Backward Compatibility</b>	<b>22</b>
<b>Example of EDI Usage</b>	<b>23</b>
Overview of EDI Payments Processing	23
Types of Information That Is Exchanged Electronically	24
Types of Electronic Payment	24
Transfer of Funds	25
Payment-Related EDI Transactions	25
<b>Acknowledgment Types</b>	<b>26</b>
TA1, Interchange Acknowledgment	26
997, Functional Acknowledgment	26
Application Acknowledgments	26
<b>Key Parts of EDI Processing Logic</b>	<b>27</b>
Structures	27
Validations, Translations, Enveloping, Acknowledgments	27
Trading Partner Agreements	27
<b>ASC X12 Version Support</b>	<b>28</b>
<b>SEF File Support</b>	<b>28</b>

---

## Chapter 3

<b>Installing the ASC X12 Manager</b>	<b>29</b>
System Requirements	29
Supported Operating Systems	29
Supported External Applications	30
Required ICAN Suite Products	30
Installing the ASC X12 Manager Composite Application	30
Increasing the Enterprise Designer Heap Size	32
Resolving Memory Errors at Enterprise Designer Startup	32
Configuring the Oracle Database	32

---

## Chapter 4

<b>Working with Validation BPs</b>	<b>34</b>
Importing Validation BPs into Projects	34
Customizing Validation BPs	35

---

Chapter 5

<b>Configuring Trading Partners</b>	<b>39</b>
Overview of the ePM Parameter Hierarchy	39
Configuring Trading Partners	40
Configuring Trading Partners	40
Setting Up Trading Partner Profiles (TPP)	44
Configuring Business Services	44
Business Service > Business Actions	45
Configuring Business Actions	45

---

Chapter 6

<b>Working with the ASC X12 Sample</b>	<b>48</b>
About the ASC X12 Manager Sample	48
Process Flow in the ASC X12 Sample	48
Process Flow in the Atlanta Environment	49
Process Flow in the Berlin Environment	50
About the X12_Host	50
Quick Steps to Get the Sample Up and Running	51
Unzipping the Sample File	53
Importing the Sample Projects	54
Understanding the 850 Feeder Project	55
About the 850 Project Connectivity Map	56
About the 850 Project BP	57
Understanding the 855 Feeder Project	58
About the 855 Project Connectivity Map	58
About the 855 Project BP	59
Configuring the Oracle External Application	61
Creating the Validation Connectivity Map	62
Creating and Activating Deployment Profiles	63
Importing and Activating Trading Partners	64
Running the X12 Sample	65
Starting the Logical Hosts	65
Preparing the Input Data	66

---

Appendix A

<b>OTD Syntax Validation BPs</b>	<b>67</b>
Activity Flow	67

## Contents

<b>Fault Handling</b>	<b>70</b>
ValidateException	70
UnmarshalException	71
GenericException	71
Other Faults	71
<b>Variables Referenced by OTD Validation BPs</b>	<b>72</b>
The Value of the <code>#{BizRespCorrPath}</code> Variable	72
<b>Index</b>	<b>75</b>

# List of Figures

Figure 1	ASC X12 Manager Validation BPs	13
Figure 2	X12 Envelope Schematic	18
Figure 3	X12 997 (Functional Acknowledgment) Segment Table	19
Figure 4	Example of a Transaction Set Header (ST)	19
Figure 5	Example of a Transaction Set Trailer (SE)	19
Figure 6	Example of a Functional Group Header (GS)	20
Figure 7	Example of a Functional Group Trailer (GE)	20
Figure 8	Example of an Interchange Header (ISA)	21
Figure 9	Example of an Interchange Trailer (IEA)	22
Figure 10	Increasing Enterprise Designer Heap Size	32
Figure 11	Validation BP .zip Files	34
Figure 12	Opening the X12 v4021 219 Full Syntax Validation Handler	36
Figure 13	Mappings for the Request/Response Correlation Key	37
Figure 14	ePM Parameter Hierarchy of TP > TP Profile > Service	40
Figure 15	Trading Partner Profile: “Properties” Tab	44
Figure 16	Business Service Configuration: “Business Actions” Tab	45
Figure 17	ASC X12 Sample Process Flow	49
Figure 18	B2B Host (X12_Host)— Project Components	51
Figure 19	The Sample Project Validation Connectivity Map	52
Figure 20	Importing Sample Projects	55
Figure 21	Connectivity Map for 850 Feeder Project	56
Figure 22	850_BP1 for the 850 Feeder Project	57
Figure 23	Connectivity Map for 855 Feeder Project	59
Figure 24	855_BP1 for the 855 Feeder Project	60
Figure 25	Locating Validation BPs	62
Figure 26	Creating the Validation Connectivity Map	63

# List of Tables

Table 1	Document Conventions	10
Table 2	Default Delimiters in X12 OTD Libraries	17
Table 3	Key Parts of EDI Processing	27
Table 4	Supported ASC X12 Versions	28
Table 5	General XDC Bindings Settings	41
Table 6	Setting ToPartner Envelope Binding Configurations	42
Table 7	Setting FromPartner DeEnvelope Binding Configurations	43
Table 8	Parameters in the (TP->TPP->) "Properties" Tab	44
Table 9	Parameters in the (TP->TPP-> Business Service) "Business Actions" Tab	45
Table 10	Deployment Profiles To Be Created and Activated	52
Table 11	X12Manager_Sample.zip Project Files	54
Table 12	Deployment Profiles To Be Created and Activated	63
Table 13	Variables Referenced by OTD Validation BPs	72



# 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

- [About This Document](#) on page 9
- [Related Documents](#) on page 10
- [References](#) on page 11
- [SeeBeyond Web Site](#) on page 11
- [SeeBeyond Documentation Feedback](#) on page 11

---

## 1.1 About This Document

This user's guide describes how to install and use the ASC X12 Manager Composite Application (ASC X12 Manager) to create ICAN Projects that process and validate X12 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 ASC X12 Manager"](#) provides an overview of the ASC X12 Manager Composite Application.
- [Chapter 3, "Installing the ASC X12 Manager"](#) describes how to install the ASC X12 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 ASC X12-specific items that can or must be configured in eXchange Partner Manager (ePM).

- **Chapter 6, “Working with the ASC X12 Sample”** describes how to import, configure, and run the ASC X12sample.
- **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 the ASC X12 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	<b>Bold</b> text	<ul style="list-style-type: none"> <li>▪ Click <b>OK</b> to save and close.</li> <li>▪ From the <b>File</b> menu, select <b>Exit</b>.</li> <li>▪ Select the <b>logicalhost.exe</b> file.</li> <li>▪ Enter the <b>timeout</b> value.</li> <li>▪ Use the <b>getClassname()</b> method.</li> <li>▪ Configure the <b>Inbound</b> File eWay.</li> </ul>
Command line arguments, code samples	Fixed font. Variables are shown in <i><b>bold italic</b></i> .	bootstrap -p <i><b>password</b></i>
Hypertext links	<b>Blue</b> text	See <b>Related Documents</b> on page 10
Hypertext links for Web addresses (URLs) or email addresses	<b>Blue underlined</b> text	<a href="http://www.seebeyond.com">http://www.seebeyond.com</a> <a href="mailto:docfeedback@seebeyond.com">docfeedback@seebeyond.com</a>

## 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 ASC X12 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*
- *ASC X12 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 Web sites provide additional information about the ASC X12 protocol:

- <http://www.disa.org>
- <http://www.x12.org/x12org/index.cfm>
- <http://www.wpc-edi.com>

---

## 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](mailto:docfeedback@seebeyond.com)

# Overview of the ASC X12 Manager

This chapter provides a general overview of the ASC X12 Manager and its place in the ICAN Suite.

## What's in This Chapter

- [About the ASC X12 Manager Composite Application](#) on page 12
- [How the ASC X12 Manager Works](#) on page 14
- [About the ASC X12 Protocol](#) on page 14
- [ASC X12 Version Support](#) on page 28
- [SEF File Support](#) on page 28

---

## 2.1 About the ASC X12 Manager Composite Application

The ASC X12 Manager Composite Application integrates with eGate Integrator, eXchange Integrator, and the ASC X12 OTD Library to enable you to design ICAN Projects that process and validate ASC X12 messages.

The ASC X12 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

### ASC X12 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 ASC X12. The ASC X12 Manager works with eXchange to provide the following during message processing:

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

### ASC X12 Manager and the ASC X12 OTD Library

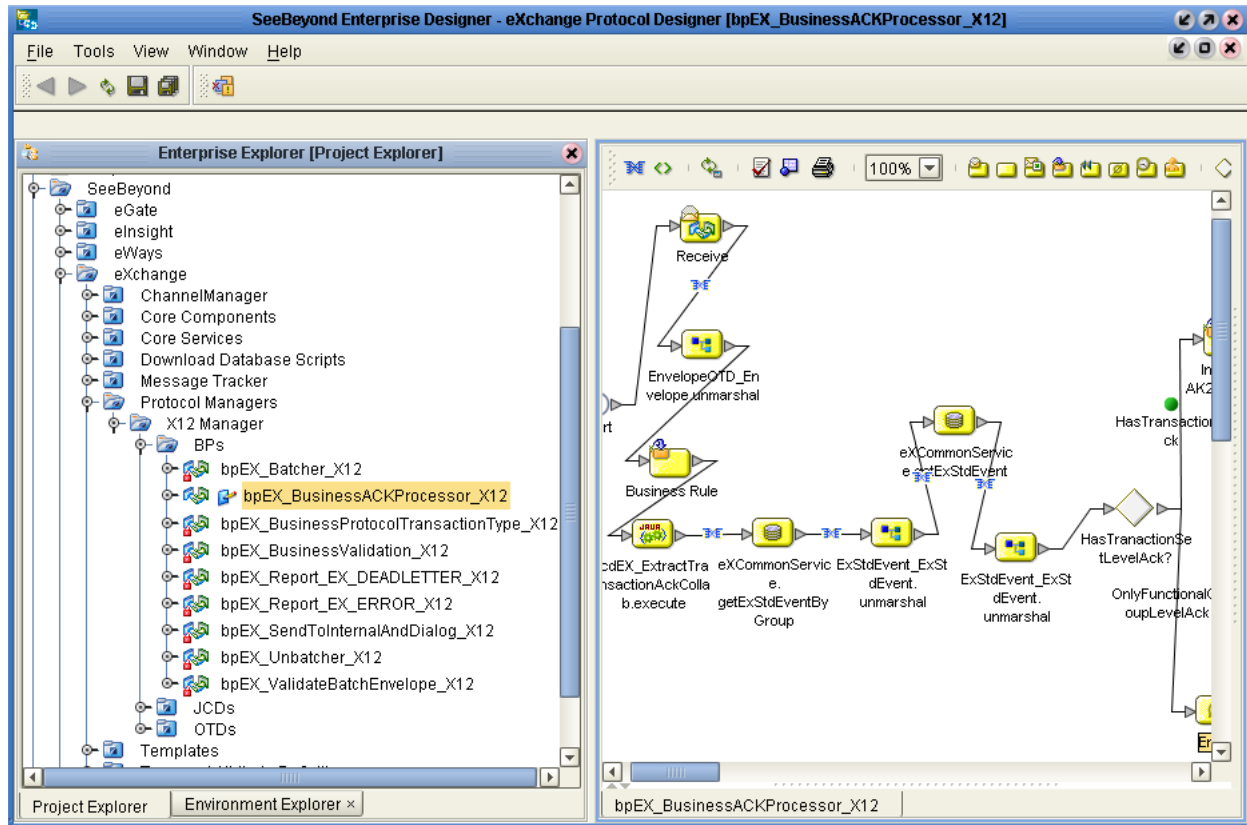
The ASC X12 Manager provides packaged Business Protocol (BP) rules to validate ASC X12 message structures, which are called Object Type Definitions (OTDs) in the ICAN Suite. The ICAN Suite provides packaged X12 OTDs with the ASC X12 OTD Library. You can also build your own OTDs with the SEF OTD wizard, which is supplied with eGate Integrator.

### Importing and customizing validation rules

The ASC X12 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 34.

The figure below shows the BPs supplied with the ASC X12 Manager and the canvas shows one of the ASC X12 BPs. All business logic is exposed, and each action is customizable.

**Figure 1** ASC X12 Manager Validation BPs



---

## 2.2 How the ASC X12 Manager Works

In the View layer: The eXchange Protocol Designer provides the ability to create ASC X12 Projects, and the eXchange Message Tracker allows searching and viewing of ASC X12 messages.

In the Services Orchestration layer: You use the eXchange Service Designer to design a transaction set; then the ASC X12 Project prepares and returns the interchange and functional acknowledgments (TA1 and 997) to the trading partner, and also performs message correlation to associate the business response to the request.

In the Integration Services layer: The ASC X12 Project validates ISA and GS envelopes from incoming messages, prepares ISA and GS envelopes for outgoing messages, batches together documents to be delivered as a single transaction (ISA), and records the activity in Message Tracking.

---

## 2.3 About the ASC X12 Protocol

This section provides the following information:

- An overview of X12, including the structure of an X12 envelope, data elements, and syntax.
- An explanation of how to use the generic message structures provided as an add-on to eGate to help you quickly create the structures you need for X12 transactions.
- An example of how X12 is used in payment processing.

### 2.3.1. What Is X12?

X12 is an EDI (electronic data interchange) standard, developed for the electronic exchange of machine-readable information between businesses.

The Accredited Standards Committee (ASC) X12 was chartered by the American National Standards Institute (ANSI) in 1979 to develop uniform standards for interindustry electronic interchange of business transactions—electronic data interchange (EDI). The result was the X12 standard.

An organization called the X12 body develops, maintains, interprets, and promotes the proper use of the ASC standard. Data Interchange Standards Association (DISA) publishes the X12 standard and the UN/EDIFACT standard. The X12 body comes together three times a year to develop and maintain EDI standards. Its main objective is to develop standards to facilitate electronic interchange relating to business transactions such as order placement and processing, shipping and receiving information, invoicing, and payment information.

X12 was originally intended to handle large batches of transactions. However, it has been extended to encompass real-time processing (transactions sent individually as they are ready to send, rather than held for batching).

### 2.3.2. What Is a Message Structure?

ASC X12 messages have a message structure that 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 ASC X12 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 transaction set 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 ASC X12 OTD for a specific version includes transaction set 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 ASC X12 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 ASC X12 transaction. The list of transactions provided is different for each version of ASC X12.

ASC X12 messages have a message structure that 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 ASC X12 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 transaction set 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 ASC X12 OTD for a specific version includes transaction set 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 ASC X12 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 ASC X12 transaction. The list of transactions provided is different for each version of ASC X12.

---

## 2.4 Components of an X12 Envelope

X12 messages are all ASCII text, with one exception: the BIN segment is binary.

Each X12 message is made up of a combination of the following elements:

- Data elements
- Segments
- Loops

Elements are separated by delimiters.

More information on each of these is provided below.

### 2.4.1. Data Elements

The data element is the smallest named unit of information in the X12 standard. Data elements can be broken down into two types. The distinction between the two is strictly a matter of how they are used. The two types are:

- Simple

If a data element occurs in a segment outside the defined boundaries of a composite data structure, it is called a *simple* data element.

- Composite

If a data element occurs as an ordinally positioned member of a composite data structure, it is called a *composite* data element. A telephone number is a simple example of a composite: It has an area code, which must precede the digits for the local exchange, which must precede the final group of digits.

Each data element has a unique reference number; it also has a name, description, data type, and minimum and maximum length.

### 2.4.2. Segments

A segment is a logical grouping of data elements. In X12, the same segment can be used for different purposes. This means that a field's meaning can change based on the segment. For example:

- The NM1 segment is for *any* name (patient, provider, organization, doctor)
- The DTP segment is for *any* date (date of birth, discharge date, coverage period)

For more information on the X12 enveloping segments, refer to [“Structure of an X12 Envelope” on page 18](#).



### 2.4.3. Loops

Loops are sets of repeating ordered segments. In X12 you can locate elements by specifying:

- The transaction set (for example, 270)
- The loop (for example, “loop 1000” or “info. receiver loop”)
- The occurrence of the loop
- The segment (for example, BGN)
- The field number (for example, 01)
- The occurrence of the segment (if it is a repeating segment)

### 2.4.4. Delimiters

In an X12 message, the various delimiters act as syntax, dividing up the different elements of a message. The delimiters used in the message are defined in the interchange control header, the outermost layer enveloping the message. For this reason, there is flexibility in the delimiters that are used.

No suggested delimiters are recommended as part of the X12 standards, but the industry-specific implementation guides do have recommended delimiters.

The default delimiters used by the SeeBeyond X12 OTD Libraries are the same as those recommended by the industry-specific implementation guides. These delimiters are shown in Table 2.

**Table 2** Default Delimiters in X12 OTD Libraries

Type of Delimiter	Default Value
Segment terminator	~ (tilde)
Data element separator	* (asterisk)
Subelement (component) separator	: (colon)
Repetition separator (version 4020 and later)	+ (plus sign)

Within eXchange Integrator, delimiters are specified at the enveloping level. The delimiters defined for an envelope apply to all transactions in the same business service. (A business service is a choreographed dialog between the two parties.)

If you do not specify delimiters, and do not override them in the payload transactions fed into FROMINTERNAL, eXchange expects the default delimiters shown in Table 2.

**Note:** *It is important to note that errors could result if the transmitted data itself includes any of the characters that have been defined as delimiters. Specifically, the existence of asterisks within transmitted application data is a known issue in X12, and can cause problems with translation.*

## 2.5 Structure of an X12 Envelope

The rules applying to the structure of an X12 envelope are very strict, to ensure the integrity of the data and the efficiency of the information exchange.

The actual X12 message structure has three main levels. From the highest to the lowest they are:

- Interchange Envelope
- Functional Group
- Transaction Set

A schematic of X12 envelopes is shown in Figure 2. Each of these levels is explained in more detail in the following sections.

**Figure 2** X12 Envelope Schematic

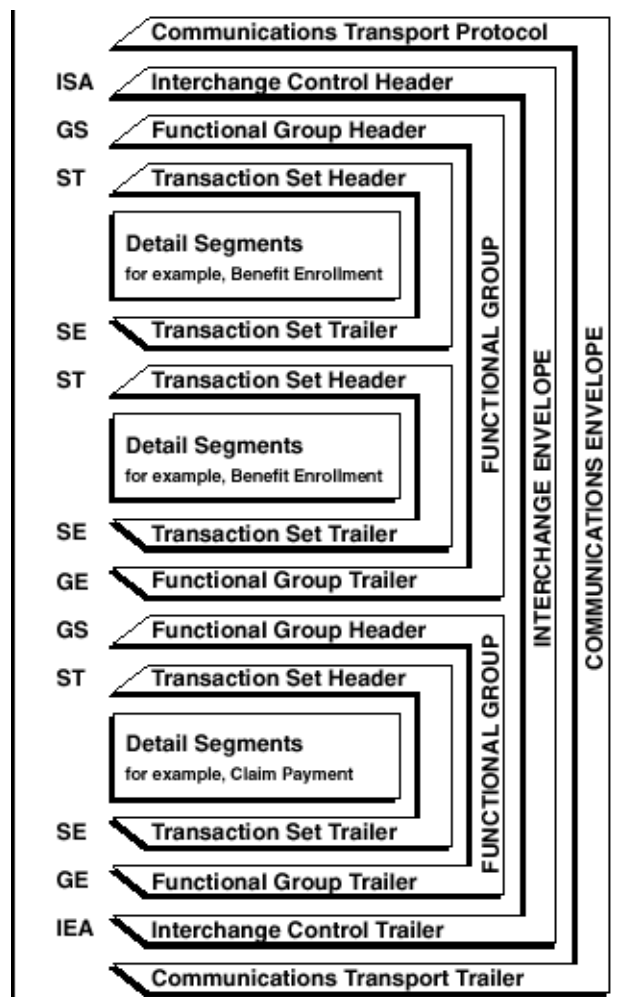


Figure 3 shows the standard segment table for an X12 997 (Functional Acknowledgment) as it appears in the X12 standard and in most industry-specific implementation guides.

**Figure 3** X12 997 (Functional Acknowledgment) Segment Table

**Table 1 - Header**

POS. #	SEG. ID	NAME	REQ. DES.	MAX USE	LOOP REPEAT
010	ST	Transaction Set Header	M	1	
020	AK1	Functional Group Response Header	M	1	
LOOP ID - AK2					999999
030	AK2	Transaction Set Response Header	O	1	
LOOP ID - AK2/AK3					999999
040	AK3	Data Segment Note	O	1	
050	AK4	Data Element Note	O	99	
060	AK5	Transaction Set Response Trailer	M	1	
070	AK9	Functional Group Response Trailer	M	1	
080	SE	Transaction Set Trailer	M	1	

### 2.5.1. Transaction Set (ST/SE)

Each transaction set (also called a transaction) contains three things:

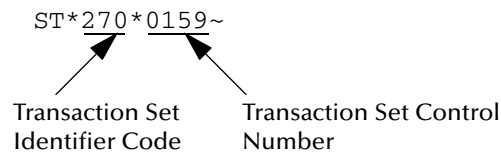
- A transaction set header
- A transaction set trailer
- A single message, enveloped within the header and footer

The transaction has a three-digit code, a text title, and a two-letter code; for example, **997, Functional Acknowledgment (FA)**.

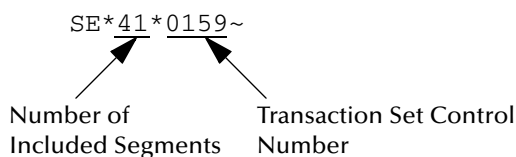
The transaction is composed of logically related pieces of information, grouped into units called segments. For example, one segment used in the transaction set might convey the address: city, state, postal code, and other geographical information. A transaction set can contain multiple segments. For example, the address segment could be used repeatedly to convey multiple sets of address information.

The X12 standard defines the sequence of segments in the transaction set and also the sequence of elements within each segment. The relationship between segments and elements could be compared to the relationship between records and fields in a database environment.

**Figure 4** Example of a Transaction Set Header (ST)



**Figure 5** Example of a Transaction Set Trailer (SE)



## 2.5.2. Functional Group (GS/GE)

A functional group is composed of one or more transaction sets, all of the same type, that can be batched together in one transmission. The functional group is defined by the header and trailer; the Functional Group Header (GS) appears at the beginning, and the Functional Group Trailer (GE) appears at the end. Many transaction sets can be included in the functional group, but all transactions must be of the same type.

Within the functional group, each transaction set is assigned a functional identifier code, which is the first data element of the header segment. The transaction sets that constitute a specific functional group are identified by this functional ID code.

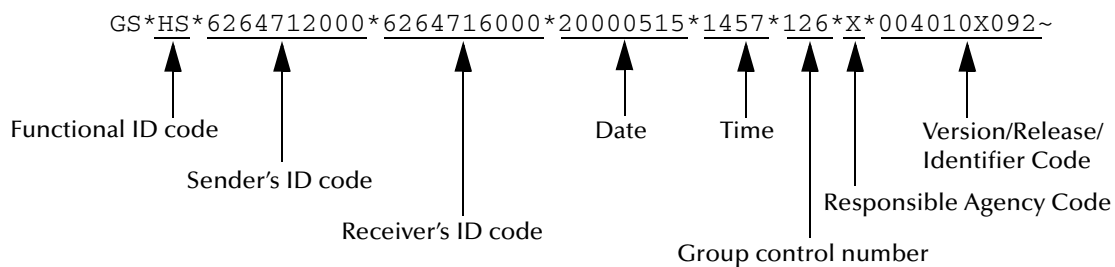
The functional group header (GS) segment contains the following information:

- Functional ID code (the two-letter transaction code; for example, PO for an 850 Purchase Order, HS for a 270 Eligibility, Coverage or Benefit Inquiry) to indicate the type of transaction in the functional group
- Identification of sender and receiver
- Control information (the functional group control numbers in the header and trailer segments must be identical)
- Date and time

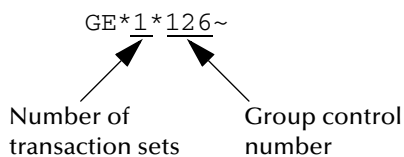
The functional group trailer (GE) segment contains the following information:

- Number of transaction sets included
- Group control number (originated and maintained by the sender)

**Figure 6** Example of a Functional Group Header (GS)



**Figure 7** Example of a Functional Group Trailer (GE)



### 2.5.3. Interchange Envelope (ISA/IEA)

The interchange envelope is the wrapper for all the data to be sent in one transmission. It can contain multiple functional groups. This means that transactions of different types can be included in the interchange envelope, with each type of transaction stored in a separate functional group.

The interchange envelope is defined by the header and trailer; the Interchange Control Header (ISA) appears at the beginning, and the Interchange Control Trailer (IEA) appears at the end.

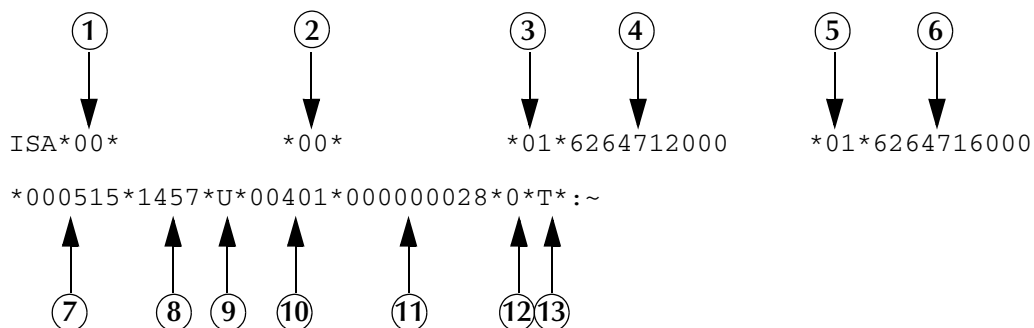
As well as enveloping one or more functional groups, the interchange header and trailer segments include the following information:

- Data element separators and data segment terminator
- Identification of sender and receiver
- Control information (used to verify that the message was correctly received)
- Authorization and security information, if applicable

The sequence of information that is transmitted is as follows:

- Interchange header
- Optional interchange-related control segments
- Actual message information, grouped by transaction type into functional groups
- Interchange trailer

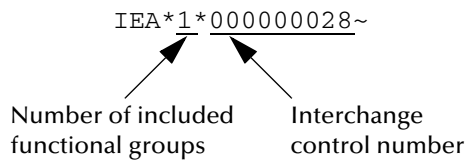
**Figure 8** Example of an Interchange Header (ISA)



Interchange Header Segments from Figure 8:

- |                                       |                                       |
|---------------------------------------|---------------------------------------|
| 1 Authorization Information Qualifier | 8 Time                                |
| 2 Security Information Qualifier      | 9 Repetition Separator                |
| 3 Interchange ID Qualifier            | 10 Interchange Control Version Number |
| 4 Interchange Sender ID               | 11 Interchange Control Number         |
| 5 Interchange ID Qualifier            | 12 Acknowledgment Requested           |
| 6 Interchange Receiver ID             | 13 Usage Indicator                    |
| 7 Date                                |                                       |

**Figure 9** Example of an Interchange Trailer (IEA)



### 2.5.4. Control Numbers

The X12 standard includes a control number for each enveloping layer:

- ISA13—Interchange Control Number
- GS06—Functional Group Control Number
- ST02—Transaction Set Control Number

The control numbers act as identifiers, useful in message identification and tracking. eXchange Integrator includes a flag for each control number, so you can choose not to assign control numbers to outgoing messages and not to store control numbers on incoming messages.

#### ISA13 (Interchange Control Number)

The ISA13 is assigned by the message sender. It must be unique for each interchange. This is the primary means used by eXchange Integrator to identify an individual interchange.

#### GS06 (Functional Group Control Number)

The GS06 is assigned by the sender. It must be unique within the Functional Group assigned by the originator for a transaction set.

eXchange ensures that the Functional Group control number GS06 in the header must be identical to the same data element in the associated Functional Group trailer, GE02.

#### ST02 (Transaction Set Control Number)

The ST02 is assigned by the sender, and is stored in the transaction set header. It must be unique within the Functional Group.

eXchange ensures that the control number in ST02 is identical with the SE02 element in the transaction set trailer, and is unique within a Functional Group (GS-GE). Once you have defined a value for SE02, eXchange Integrator uses the same value for SE02.

---

## 2.6 Backward Compatibility

Each version of X12 is slightly different. Each new version has some new transactions; in addition, existing transactions can change from version to version.

New versions of X12 are usually backward compatible; however, this is not a requirement of the X12 rules. You should not expect different versions of X12 to be backward compatible, but you can expect that when you analyze the differences only a few changes are required in the message structures.

**Note:** *In this context, backward compatible means that software that parses one version can, in some circumstances, be unable to parse the next version, even if the software ignores any unexpected new segments, data elements at the end of segments, and sub-elements at the end of composite data elements. Not backward compatible means that required segments can disappear entirely, data elements can change format and usage, and required data elements can become optional.*

---

## 2.7 Example of EDI Usage

This section provides an overview of the normal processes involved in EDI payment processing.

**Note:** *This section is a general overview of how electronic payments processing is used. Not everything in this section applies to the use of X12 in processing payments.*

### 2.7.1. Overview of EDI Payments Processing

EDI payments processing encompasses both collection and disbursement transactions. The exchange of funds is accomplished by means of credit and debit transfers. It can also include a related bank balance, as well as transaction and account analysis reporting mechanisms.

Most non-monetary EDI trading partner communications are handled either directly between the parties or indirectly through their respective value added networks (VANs). However, the exchange of funds requires a financial intermediary. This is normally the bank or banks that hold deposit accounts of the two parties.

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. The remittance information, which acts as an electronic check stub, can be sent in any of the following ways:

- Directly between trading partners or through their respective EDI VAN mailboxes
- Through the banking system, with the beneficiary's bank sending notice of payment to the beneficiary
- By the originator to the originator's bank as an order to pay, with the originator's bank notifying the beneficiary

The trading partners and the capabilities of their respective banks determine the following:

- The routing of the electronic check stub

- Which of the following the payment is:
  - ♦ a debit authorized by the payor and originated by the beneficiary
  - ♦ a credit transfer originated by the payor

## Types of Information That Is Exchanged Electronically

There are several types of information that can be exchanged electronically between bank and customer, including:

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

The electronic payment mechanism, which is a subset of EDI, involves two separate activities:

- The exchange of payment orders, causing value to transfer from one account to another
- The exchange of related remittance information in standardized machine-processable formats.

## Types of Electronic Payment

The electronic payment can be either of the following:

- Credit transfer, initiated by the payor
- 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 in the form of an X12 Payment Order/Remittance Advice (transaction set 820).

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.

A corporate-to-corporate payment performs two functions:

- Transfers actual monetary value
- Transfers notification of payment from payor to payee

When a credit transfer occurs, these two functions are sometimes treated as one, and sometimes treated separately. The two functions can travel in either of these two ways:

- Together through the banking system
- Separately and by different routes



X12 820 is a data format for transporting a payment order from the originator to its bank. This payment order can be either of the following:

- An instruction to the originator's bank to originate a credit transfer
- An instruction to the trading partner to originate a debit transfer against the payor's bank account

Once this decision has been made, the 820 transports the remittance information to the beneficiary. The transfer can either be through the banking system or by a route that is separate from the transport of funds.

**Note:** *Whenever the 820 remittance information is not transferred with the funds, it can be transmitted directly from the originator to the beneficiary. It can also be transmitted through an intermediary, such as a VAN.*

## Transfer of Funds

Before funds can be applied against an open accounts receivable account, the beneficiary must reconcile the two streams—the payment advice from the receiving bank and the remittance information received through a separate channel—that 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 might have problems balancing the accounts receivable ledger.

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 amount to the beneficiary's bank for credit to the beneficiary's account.

If the payment order specifies a debit transfer, the originator is the beneficiary. In this case, the beneficiary's bank originates the value transfer, and the payor's account is debited (charged) for a set amount, which is credited to the originator's (beneficiary's) bank account. The payor must issue approval to its bank to honor the debit transfer, either before the beneficiary presents the debit transfer or at the same time. This debit authorization or approval can take one of four 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

### 2.7.2. Payment-Related EDI Transactions

X12 uses an end-to-end method to route the 820 Payment Order/Remittance Advice from the originator company through the banks to the beneficiary. This means that there can be several relay points between the sender and the receiver.

The 820 is wrapped in an ACH banking transaction for the actual funds transfer between the banks.

---

## 2.8 Acknowledgment Types

X12 includes two types of acknowledgment, the TA1 Interchange Acknowledgment and the 997 Functional Acknowledgment.

### 2.8.1. TA1, Interchange Acknowledgment

The TA1 acknowledgment verifies the interchange envelopes only. The TA1 is a single segment and is unique in the sense that this single segment is transmitted without the GS/GE envelope structures. A TA1 acknowledgment can be included in an interchange with other functional groups and transactions.

### 2.8.2. 997, Functional Acknowledgment

The 997 includes much more information than the TA1; see [Figure 3 on page 19](#). The 997 was designed to allow trading partners to establish a comprehensive control function as part of the business exchange process.

There is a one-to-one correspondence between a 997 and a functional group. Segments within the 997 identify whether the functional group was accepted or rejected. Data elements that are incorrect can also be identified.

Many EDI implementations have incorporated the acknowledgment process into all of their electronic communications. Typically, the 997 is used as a functional acknowledgment to a functional group that was transmitted previously.

The 997 is the acknowledgment transaction recommended by X12.

The acknowledgment of the receipt of a payment order is an important issue. Most corporate originators want to receive at least a Functional Acknowledgment (997) from the beneficiary of the payment. The 997 is created using the data about the identity and address of the originator found in the ISA and/or GS segments.

### 2.8.3. Application Acknowledgments

Application acknowledgments are responses sent from the destination system back to the originating system, acknowledging that the transaction has been successfully or unsuccessfully completed. The application advice (824) is a generic application acknowledgment that can be used in response to any X12 transaction. However, it has to be set up as a response transaction; only TA1 and 997 transactions are sent out automatically.

Other types of responses from the destination system to the originating system, which can also be considered application acknowledgments, are responses to query transactions—for example, the Eligibility Response (271) is a response to the Eligibility Inquiry (270).

## 2.9 Key Parts of EDI Processing Logic

The five key parts of EDI processing logic are listed in Table 3.

**Table 3** Key Parts of EDI Processing

Term	Description	Language Analogy	eGate Component
structures	format, segments, loops	syntax rules	OTD elements and fields
validations	data contents “edit” rules	semantic rules	validation methods
translations (also called mappings)	reformatting or conversion	translation	collaborations
enveloping	header and trailer segments	envelope for a written letter	the special “envelope” OTDs: FunctionalGroupEnv and InterchangeEnv
acks	acknowledgments	return receipt	specific acknowledgment elements in the OTD

eGate uses the structures, validations, translations, enveloping, and acknowledgments listed below to support the X12 standard.

### 2.9.1. Structures

The X12 OTD Library includes pre-built OTDs for all supported X12 versions. These OTDs can be viewed in the OTD Editor, but cannot be modified.

To customize the OTD structure—for example, to add a segment or loop—you must first generate a SEF file (typically using a third-party tool, such as the EDISIM tool from Foresight Corporation). You then use the SEF OTD Wizard to generate the OTD.

### 2.9.2. Validations, Translations, Enveloping, Acknowledgments

Within each OTD are Java methods and Java bean nodes for handling validation; and the marshal and unmarshal methods of the two **envelope** OTDs handle enveloping and de-enveloping. No pre-built translations are supplied with the OTD libraries; these can be built in an eGate GUI called the Java Collaboration Editor (JCE).

*Note:* In eGate, X12 translations are called Collaborations.

### 2.9.3. Trading Partner Agreements

There are three levels of information that guide the final format of a specific transaction. These three levels are:

- The X12 standard—The Accredited Standards Committee publishes a standard structure for each X12 transaction.

- Industry-specific Implementation Guides—Specific industries publish Implementation Guides customized for that industry. Normally, these are provided as recommendations only. However, in certain cases, it is extremely important to follow these guidelines. Specifically, since HIPAA regulations are law, it is important to follow the guidelines for these transactions closely.
- Trading Partner Agreements—It is normal for trading partners to have individual agreements that supplement the standard guides. The specific processing of the transactions in each trading partner’s individual system can vary from one site to another. Because of this, additional documentation providing information about the differences is helpful to the site’s trading partners and simplifies implementation. For example: Although a certain code might be valid in an implementation guide, a specific trading partner might not use that code in transactions; in such a case, it would be important to include that information in a trading partner agreement.

---

## 2.10 ASC X12 Version Support

This product provides support for the following ASC X12 versions:

**Table 4** Supported ASC X12 Versions

▪ 4010	▪ 4020	▪ 4030	▪ 4040	▪ 4050	▪ 4060
▪ 4011	▪ 4021	▪ 4031	▪ 4041	▪ 4051	▪ 4061
▪ 4012	▪ 4022	▪ 4032	▪ 4042	▪ 4052	

---

## 2.11 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.

# Installing the ASC X12 Manager

This chapter describes how to install the ASC X12 Manager Composite Application, its documentation, and sample Projects. This chapter also includes the system requirements and supported operating systems for the ASC X12 Manager Composite Application.

## What's in This Chapter

- [System Requirements](#) on page 29
- [Supported Operating Systems](#) on page 29
- [Supported External Applications](#) on page 30
- [Required ICAN Suite Products](#) on page 30
- [Installing the ASC X12 Manager Composite Application](#) on page 30
- [Increasing the Enterprise Designer Heap Size](#) on page 32
- [Configuring the Oracle Database](#) on page 32

---

## 3.1 System Requirements

Each ASC X12 validation BP .sar file requires approximately 6 MB disk space. The large size of the ASC X12 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 32](#).

Other than that, the system requirements for the ASC X12 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 ASC X12 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 x86) and Linux Advanced Server 2.1 (Intel x86)

---

### 3.3 Supported External Applications

Database support for the ASC X12 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 ASC X12 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 ASC X12 Manager Composite Application

During the ASC X12 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 ASC X12 Manager. You must have uploaded a **license.sar** to the ICAN Repository that includes a license for the ASC X12 Manager.

## To install the ASC X12 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 **X12\_Manager.sar** (to install ASC X12 Manager)
  - E **ASC\_X12\_OTD\_Lib\_\*\*\*.sar** (to install the ASC X12 OTDs as described in the *ASC X12OTD Library User's Guide*)
  - F **ASC\_X12\_OTD\_Validation\_BP\_\*\*\*.sar** (to install the BPs to validate the OTDs installed in step D)
- 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 ASC X12 Sample” on page 48**, do the following:
  - A Upload the following items:
    - ♦ **ASC\_X12\_OTD\_Lib\_v4010sar** (to install the ASC X12 OTDs)
    - ♦ **ASC\_X12\_OTD\_Validation\_BP\_v4010.sar** (to install the validation BP)
    - ♦ **HIPAA\_2000\_Addenda\_OTD\_Lib.sar** (optional)
    - ♦ **HIPAA\_2000\_Addenda\_OTD\_Validation\_BP.sar** (optional)
    - ♦ **FileeWay.sar** (to install the File eWay)
    - ♦ **HTTPeWay.sar** (to install the HTTP(S) eWay)
    - ♦ **X12\_ManagerDocs.sar** (to install the user's guide and the sample)
  - B In the Enterprise Manager, click the **DOCUMENTATION** page, and click **ASC X12 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**.

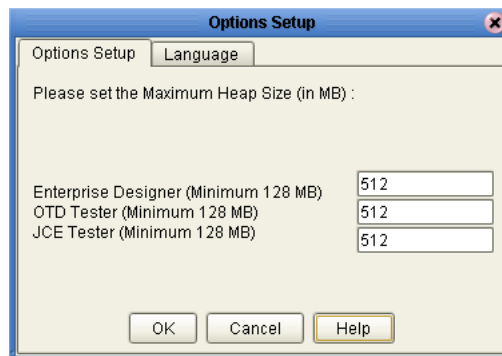
## 3.6 Increasing the Enterprise Designer Heap Size

Due to the size of the ASC X12 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 10** 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 ASC X12 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 ASC X12 Manager, the database must contain the following usernames and passwords:

- user: **ex\_A**, password: **ex\_A**



- user: **ex\_B**, password: **ex\_B**

# Working with Validation BPs

The ASC X12 Manager provides packaged BPs that validate corresponding X12 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

- [Importing Validation BPs into Projects](#) on page 34
- [Customizing Validation BPs](#) on page 35

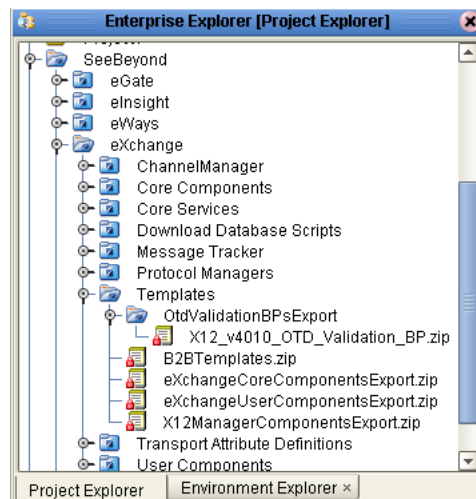
## 4.1 Importing Validation BPs into Projects

When you install a validation BP as described in [Installing the ASC X12 Manager Composite Application](#) on page 30, 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 11** 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.

---

## 4.2 Customizing Validation BPs

In the following procedures, you open an ASC X12 syntax-validation BP and locate two activities that involve a correlation key in a Request/Reply pair of transactions. For each of those activities, you locate the OTD node that currently supplies data to the correlation key, and change it to a different OTD node.

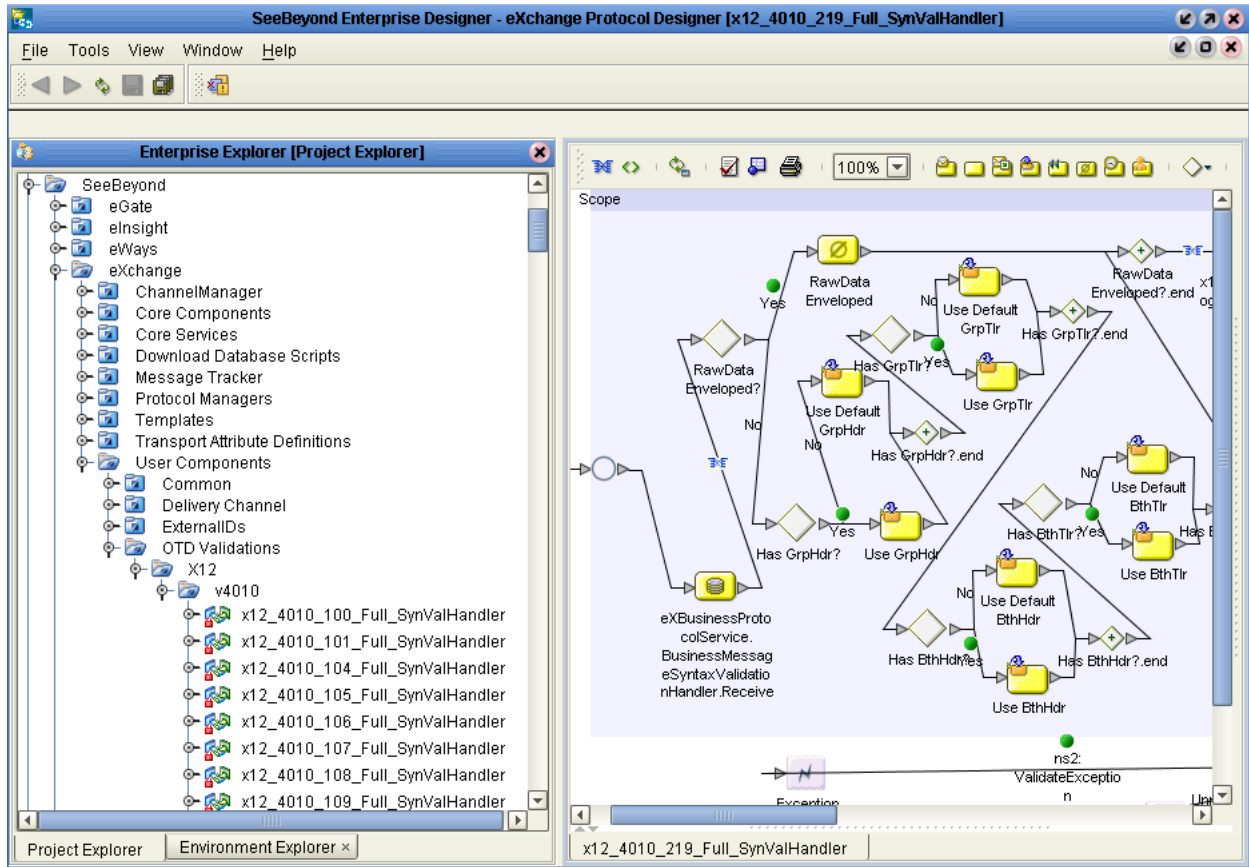
For more in-depth information on the activity flow, fault-handling, and variable usage in syntax-validation handler BPs, see [Appendix A](#).


It is assumed you have already installed eXchange and the ASC X12 Manager, as well as at least one ASC X12 OTD library and a validation BP corresponding to it. This procedure uses version 4021; if you use a different version, adjust accordingly.

To customize a validation handler BP so as to use a specific correlation key

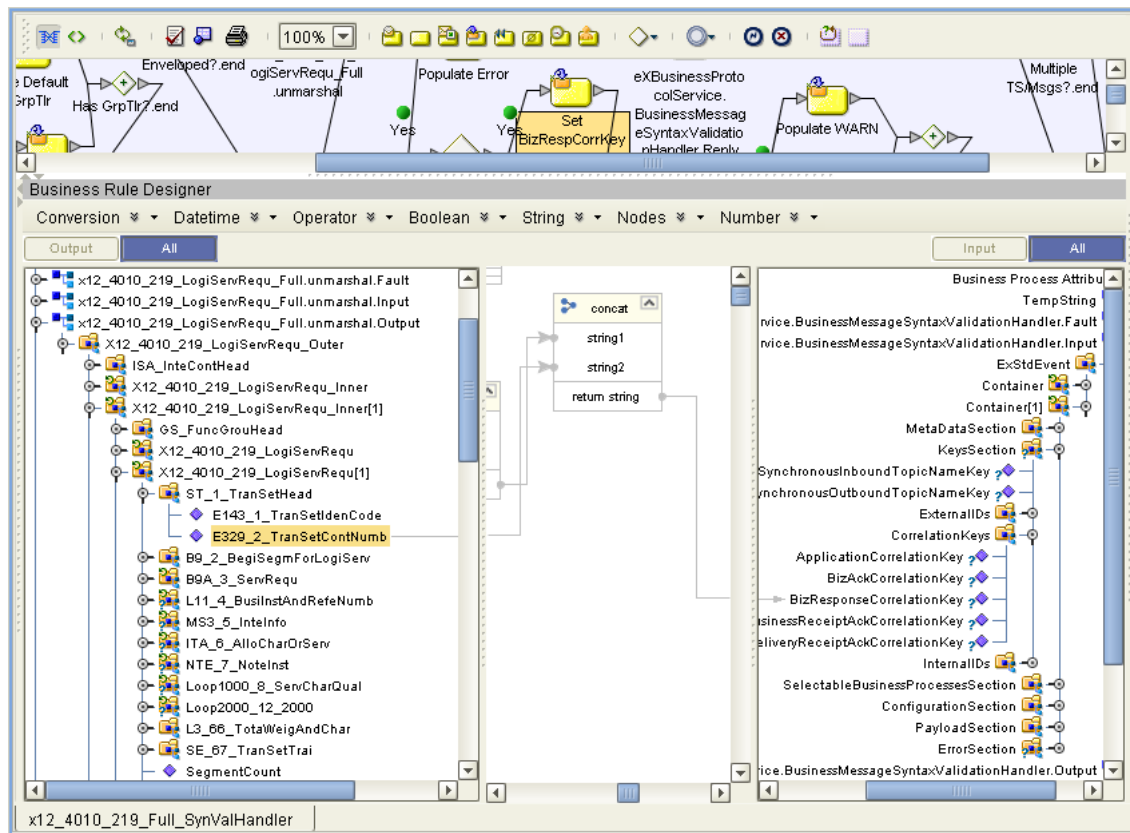
- 1 In the **Project Explorer** tab in the Enterprise Designers, expand the **SeeBeyond > eXchange > User Components > OTD Validations > X12 > v4021** folders.
- 2 Right-click **X12\_4021\_219\_Full\_SynValHandler**, click **Check Out**, and expand the folder.

Figure 12 Opening the X12 v4021 219 Full Syntax Validation Handler



- 3 On the canvas toolbar, click  to open the Business Rule Designer (BRD).
- 4 On the canvas, click the **Set BizRespCorrKey using BizTxID** business rule.
- 5 Expand the BRD pane and trace the mappings without changing them.

**Figure 13** Mappings for the Request/Response Correlation Key



- ◆ The Input side of the mapping receives the end result of a multiple concatenation into a node named **BizResponseCorrelationKey**, which is located under:
  - eXBusinessProtocolService.BusinessMessageSyntaxValidationHandler.Input
  - ExStdEvent
  - Container[1]
  - KeysSection
  - CorrelationKeys
- ◆ Several nodes on the left side go into this mapping, but you are only interested in the one that feeds into the final (rightmost) **concat** operation: Its name is **E329\_2\_TransSetContNumb**, and it is the second node located under:
  - X12\_4021\_219\_LogiServRequ\_Outer
  - X12\_4021\_219Logi\_Serv\_Requ\_Inner[1]
  - X12\_4021\_219\_LogiServRequ[1]
  - ST\_1\_1\_TransSetHead

In Figure 13, the highlighted mapping is the one you will replace.

- 6 Delete the line connecting **E329\_2\_TransSetContNumb** to **string2** of the **concat** box.
- 7 Replace it with a mapping from a node that is more suitable as a business correlation key, such as **B9A\_3\_ServRequ/E1644\_1\_ServRequCode**.
- 8 Validate the BP and save your changes.

- 9 Perform the same steps for the business rule for the “NotRequest” path in the BP:  
**Set BizRespCorrKey using ProtRespToMsgID**
- 10 Validate the BP and save your changes.

Instead of using the value passed by **E329\_2\_TransSetContNumb** as a business correlation key, this BP now uses the value passed by the node you selected, such as **E1644\_1\_ServRequCode**.

# Configuring Trading Partners

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

## What's in This Chapter

- [Overview of the ePM Parameter Hierarchy](#) on page 39
- [Configuring Trading Partners](#) on page 40
- [Setting Up Trading Partner Profiles \(TPP\)](#) on page 44
- [Configuring Business Services](#) on page 44
- [Configuring Business Actions](#) on page 45

---

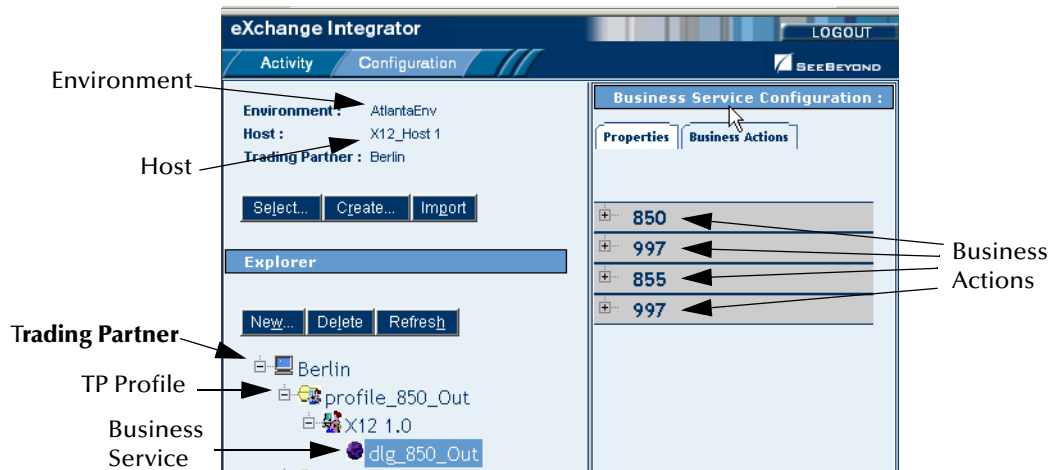
## 5.1 Overview of the ePM Parameter Hierarchy

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).

Figure 14 shows an Environment (AtlantaEnv) with one TP (**Berlin**), whose first TPP (**profile\_850\_Out**) is fully expanded; under the BAD (ASC X12 version 1.0) is its "leaf" terminus—a business service (**dlg\_850\_Out**) with four actions (850, 997, 855, 997).

**Figure 14** ePM Parameter Hierarchy of TP > TP Profile > Service



## 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 ASC X12 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 5** 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 6** 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 <b>cron</b> syntax and consists of six (or optionally seven) arguments, separated by spaces, to specify: <b>second, minute, hour, day-of-month, month, day-of-week</b> (and optionally <b>year</b> ). 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 ASC X12: bpEX_Batcher_ASC X12, used in the sample.)
ISA01 Author Info Qual	req	Refer to the ASC X12 documentation.
ISA02 Author Information		Refer to the ASC X12 documentation.
ISA03 Sec Info Qual	req	Refer to the ASC X12 documentation.
ISA04 Security Information		Refer to the ASC X12 documentation.
ISA05 IC Sender ID Qual	req	Refer to the ASC X12 documentation.
ISA06 Interchange Sender ID		Refer to the ASC X12 documentation.
ISA07 IC Rcvr ID Qual	req	Refer to the ASC X12 documentation.
ISA08 Interchange Rcvr ID	req	Refer to the ASC X12 documentation.
ISA11 IC Control Standard Identifier	U	Refer to the ASC X12 documentation.
ISA12 IC Version Number	00401	Refer to the ASC X12 documentation.

**Table 6** Setting ToPartner Envelope Binding Configurations

Parameter Name	Default Value	Description / Notes
ISA13 IC Control Number	0	Refer to the ASC X12 documentation.
ISA14 Acknowledgment Requested	1	Refer to the ASC X12 documentation.
ISA15 Usage Indicator	P	Refer to the ASC X12 documentation.
ISA16 Comp Elem Sep	:	If you use <i>nondefault</i> delimiters (for example, if you use “!” for segment terminator in v4060), you must ensure that your business rules manually pass the nondefault delimiters into the <b>ExStdEvent/PayloadSection/Envelopes/BusinessProtocol/</b> location: In other words, pass the ISA into <b>.../Batch/Header</b> , the IEA into <b>.../Batch/Trailer</b> , the GS into <b>.../Group/Header</b> , and the GE into <b>.../Group/Trailer</b> . To use a control character as a delimiter, pass the escaped Unicode UTF-16 representation of the character (\uXXXX). For example, if you wanted to use a carriage return (ASCII 0x0d) as a delimiter, you would pass the string <b>\u000d</b>
Segment Terminator	~	
Element Separator	*	
GS06 Group Control Num	5	

**Table 7** Setting FromPartner DeEnvelope Binding Configurations

Parameter Name	Default Value	Description / Notes
Batch Splitter Handler	[None]	Select a BP from the list.
ISA01 Author Info Qual	req	Refer to the ASC X12 documentation.
ISA02 Author Information		Refer to the ASC X12 documentation.
ISA03 Sec Info Qual	req	Refer to the ASC X12 documentation.
ISA04 Security Information		Refer to the ASC X12 documentation.
ISA05 IC Sender ID Qual	req	Refer to the ASC X12 documentation.
ISA06 Interchange Sender ID		Refer to the ASC X12 documentation.
ISA07 IC Rcvr ID Qual	req	Refer to the ASC X12 documentation.
ISA08 Interchange Rcvr ID		Refer to the ASC X12 documentation.
ISA11 IC Control Standard Identifier	U	Refer to the ASC X12 documentation.
ISA12 IC Version Number	00401	Refer to the ASC X12 documentation.
ISA14 Acknowledgment Requested	1	Refer to the ASC X12 documentation.

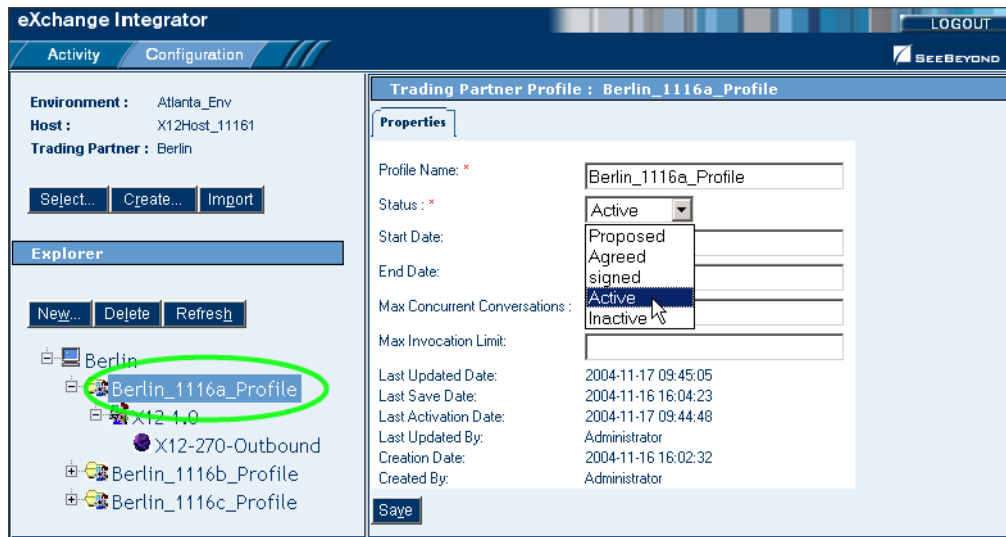
**Table 7** Setting FromPartner DeEnvelope Binding Configurations (Continued)

Parameter Name	Default Value	Description / Notes
ISA15 Usage Indicator	P	Refer to the ASC X12 documentation.

## 5.3 Setting Up Trading Partner Profiles (TPP)

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

**Figure 15** Trading Partner Profile: “Properties” Tab



**Table 8** 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

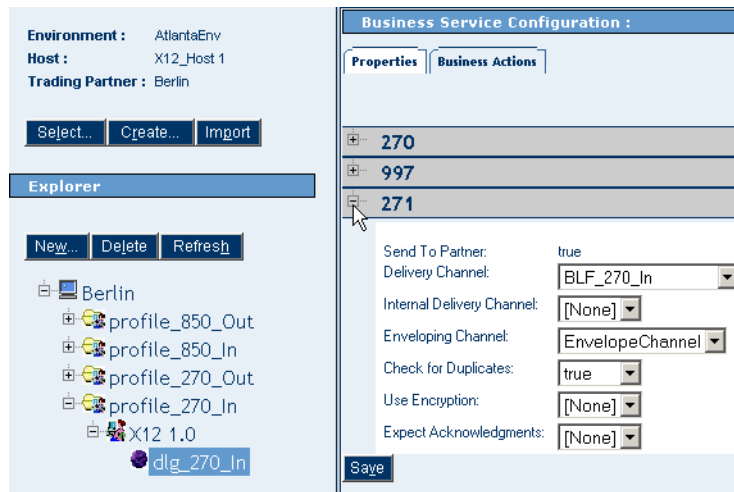
Configuration parameters for a business service (or messaging service) are organized into two major tabs: **Properties** and **Components**.

### 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 9.

## 5.5 Configuring Business Actions

**Figure 16** Business Service Configuration: “Business Actions” Tab



**Table 9** 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)	<b>true</b> indicates an <i>outbound</i> (Send ToPartner) action. <b>false</b> indicates an <i>inbound</i> (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.
Enveloping Channel	(dropdown)	Choose from a dropdown list of XDC bindings that have been defined for this TP.
Expect Acknowledgments	[None]	If set to <b>true</b> , then an error is issued if no CONTRL is received within the Time-Out period (see below). The value set here – either <b>true</b> or <b>false</b> – appears in Message Tracking of Request/Response.
Character Set Encoding		(Leave blank for default character set encoding.)
Send warnings with ACKs	[None]	

**Table 9** Parameters in the (TP->TPP-> Business Service) “Business Actions” Tab (Continued)

Parameter Name	Default Value	Description / Notes
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 997L. 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.
GS01 Functional ID	(depends on business action)	This must match the value of the “Group Name” attribute of this business action, set in the B2B host’s business service. For example, a Group Name of “HS” is for a 270 action; “HB” for a 271; “PO” for an 850; “PR” for an 855; “FA” for a 997.
GS02 Application Sender Code	req	Refer to the X12 standards documentation.
GS03 Application Rcvr Code	req	Refer to the X12 standards documentation.
GS04 Date Format	CCYYMMDD	Choose from the list to use four-digit or two-digit year format. Examples: <ul style="list-style-type: none"> <li>▪ 20041201 = December 1, 2004</li> <li>▪ 050112 = January 12, 2004</li> </ul>

**Table 9** Parameters in the (TP->TPP-> Business Service) “Business Actions” Tab (Continued)

Parameter Name	Default Value	Description / Notes
GS05 Time Format	HHMM	Choose from the list to specify seconds and degree of accuracy. Examples: <ul style="list-style-type: none"> <li>▪ 2359 = 11:59PM</li> <li>▪ 235959 = 11:59:59PM</li> <li>▪ 23595999 = 11:59:59.99PM</li> </ul>
GS07 Resp Agency Code	X	Refer to the X12 standards documentation.
GS08 Vers/Rel/Indust ID Code	req	Refer to the X12 standards documentation.
Starting Control Number	0	

# Working with the ASC X12 Sample

The ASC X12 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 ASC X12 Projects and business logic.

## What's in This Chapter

- [About the ASC X12 Manager Sample](#) on page 48
- [Quick Steps to Get the Sample Up and Running](#) on page 51
- [Unzipping the Sample File](#) on page 53
- [Importing the Sample Projects](#) on page 54
- [Understanding the 850 Feeder Project](#) on page 55
- [Configuring the Oracle External Application](#) on page 61
- [Creating and Activating Deployment Profiles](#) on page 63
- [Importing and Activating Trading Partners](#) on page 64
- [Running the X12 Sample](#) on page 65

---

## 6.1 About the ASC X12 Manager Sample

The ASC X12 Manager sample includes several Projects that you can import into Enterprise Designer to see how ICAN Projects for ASC X12 are designed.

The sample demonstrates how ASC X12 Manager is used for outbound and inbound message processing, exchanging ASC X12 850 requests and 855 responses with 997 acknowledgments between enterprises named “Atlanta” and “Berlin”.

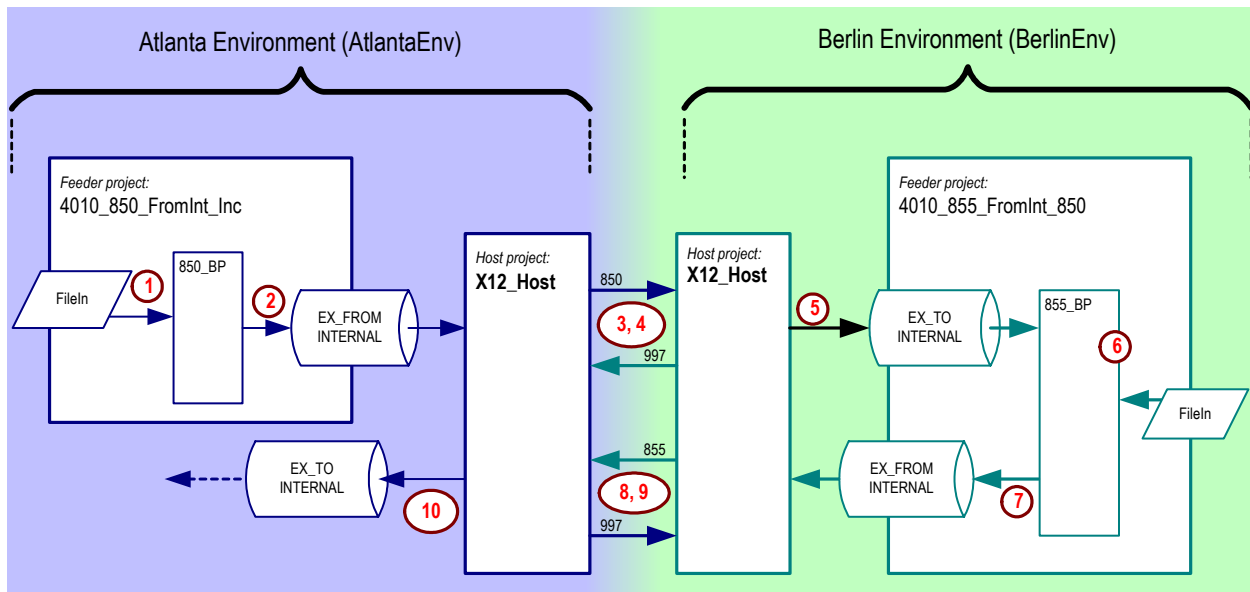
The following section describes the sample process flow in detail.

### 6.1.1 Process Flow in the ASC X12 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 17 ASC X12 Sample Process Flow



## Process Flow in the Atlanta Environment

### Initiate processing and prepare/pass the 850 action

- 1 The Atlanta feeder Project, **4010\_850\_FromInt\_Inc**, is triggered by an inbound **File** eWay when it detects a file of the form **X12\_dlg\_850\_Out\*.in**. It passes XML data to the **850\_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\_850\_Out**, the action is **850**, and the data comes from a well-qualified filename.)
- 2 Using these four data items, the **850\_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 **X12\_Host**.
  - ♦ For more information about **ExStdEvent**, standard eXchange JMS topics like **EX\_FROMINTERNAL**, and core BPs like **bpEX\_MainFromTP**, see the *eXchange Integrator Protocol Designer's Guide*.
  - ♦ For more information about the **850\_BP**, see ["About the 850 Project BP" on page 57](#).
- 3 Based on the specified service and action (**dlg\_850\_Out** and **850**) in the **ExStdEvent** message, the ASC X12 Manager and eXchange process the message, and then the Atlanta host passes an **855** response to the specified trading partner.

## Process Flow in the Berlin Environment

### Validate the 850 request, acknowledge, and prepare/pass the 855

- 1 When Berlin's B2B host (also named **X12\_Host**) receives the inbound 850 request from its trading partner Atlanta, it validates the message (per a business message syntax validation selector/handler) and sends back a 997 acknowledgment.
- 2 One of **X12\_Host**'s BPs (**bpEX\_SendToInternalAndDialogX12**) publishes the **ExStdEvent** message to a standard eXchange topic named **EX\_TOINTERNAL**, one of whose subscribers is a sample process named **855\_BP**
- 3 The **855\_BP** process, having obtained the **ExStdEvent** message, uses the Batch eWay to read information to generate a response. For more information on **855\_BP**, see ["About the 855 Project BP" on page 59](#).
- 4 The **855\_BP** process creates a new **ExStdEvent** message containing the 855 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, **X12\_Host**.
- 5 Based on the specified service and action (**dlg\_850\_In** and **855**) that it finds in the **ExStdEvent**, the Berlin host passes an 855 response to the specified trading partner (**Atlanta**).

### On the Atlanta side: Validate the 855 response, acknowledge, and continue

- 6 When Atlanta's B2B host (**X12\_Host**) receives the inbound 855 request from its trading partner Berlin, it validates the message (per a business message syntax validation handler) and sends back a 997 acknowledgment.

One of the **X12\_Host**'s BPs (**bpEX\_SendToInternalAndDialogX12**) publishes the message to the standard eXchange topic **EX\_TOINTERNAL**.

## 6.1.2 About the X12\_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 18 on page 51](#)): It starts with an outbound internal-to-host action, then continues with an outbound-to-TP action (such as an 850 or a 270) and a 997 acknowledgment, followed by an inbound-from-TP reply (such as an 855 or a 271) and a 997 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 a 997 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 a 997 acknowledgment.
- **mad\_X12** is the host project's message attributes definition (MAD).
- **X12** 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 “X12”.
- ♦ This window also references the MAD, **mad\_X12**. To find it, open the MAD folder.
- ♦ Its External Delivery Channels window defines two delivery channels for the ASC X12 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\_X12\_Host** is the map of which activation creates the “eXchange Service” external:
  - ♦ Its only input is an instance of **X12\_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 18** B2B Host (**X12\_Host**)— Project Components

---

## 6.2 Quick Steps to Get the Sample Up and Running

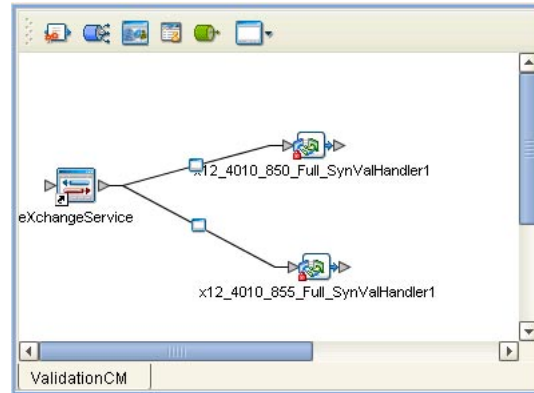
The following provides an overview of the steps necessary to get the sample up and running as quickly as possible. For detailed information, refer to the cross reference mentioned for each step.

To configure and run the ASC X12 Manager sample:

- 1 Unzip the ASC X12 Manager sample file in the following folder as described in [“Unzipping the Sample File” on page 53](#).  
**C:\temp\exChange**
- 2 Import the sample Project .zip file from the following folders as described in [“Importing the Sample Projects” on page 54](#).  
**C:\temp\exChange\Sample\X12\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 61](#).
- 4 In the **exChange > Deployment** folder, create a Connectivity Map to specify which validation BPs are to be used by the Project as described in [“Creating the Validation Connectivity Map” on page 62](#).

The validation BPs reside in the **SeeBeyond > eXchange > User Components > OTD Validations > X12 > v4010** folder. Set the eWay icon to **Inbound Exchange Service**. The figure below shows the completed Connectivity Map.

**Figure 19** The Sample Project Validation Connectivity Map



- 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 63](#).

**Table 10** Deployment Profiles To Be Created and Activated

Project Name	Deployment Profile(s)	Extra Steps
X12_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 X12_Host eXchange Service for each profile.
Errors_to_File	2 profiles; one for AtlantaEnv and a second for BerlinEnv	Map to the <b>Directory</b> constant in each Environment before activating profiles.
4010_850_FromInt_Inc	1 profile for AtlantaEnv	Map to the <b>X12_DataDir</b> constant in the Environment before activating the profile.
4010_855_FromInt_850	1 profile for BerlinEnv	Map to the <b>X12_DataDir</b> constant in the Environment before activating the profile.
eXchange > Deployment	2 profiles; one for AtlantaEnv and a second for BerlinEnv	--

If the error “Failed to obtain JNDI name prefix(ERROR\_GET\_JNDINAMEPREFIX)” occurs, you need to map to the **X12\_DataDir** constant.

- 6 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 64](#).
  - A In the Configuration tab, click **Import**.
  - B Open the Repository and **BerlinEnv** and click **X12\_Host1**.
  - C Enter **Atlanta** for the name, browse to **C:\temp\exChange\Sample\TradingPartners\X12TP\_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\X12TP\_for-AtlantaEnv\_to-from\_Berlin.xml**.
- 7 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 X12 Sample” on page 65](#).
- 8 Rename the *~in* extension to **.in** in the **c:\temp\exChange\Sample\X12\Data\Atlanta** folder.

---

## 6.3 Unzipping the Sample File

The ASC X12 Manager sample Projects are included in the **X12ManagerDocs.sar**. This file is uploaded separately from the ASC X12 Manager product file (**X12Manager.sar**) during installation. For information about uploading the **X12ManagerDocs.sar**, refer to [Installing the ASC X12 Manager Composite Application](#) on page 30.

Once you have uploaded the **X12ManagerDocs.sar** to the Repository and you have downloaded the sample Project (**X12Manager\_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 **X12dlg\_\***.*

Unzip the **X12Manager\_Sample.zip** file in the **c:\temp\exChange** folder. The folder now contains the data files and importable Projects that make up the X12 sample.

The Projects are in the following folders:

- **c:\temp\exChange\Sample\Common\Projects**  
(eXchange Project and Environment)
- **c:\temp\exChange\Sample\X12\Projects**  
(ASC X12 Projects)

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

**Table 11 X12Manager\_Sample.zip Project Files**

Project File Name	Description
<b>eX_Common_Projects.zip</b>	eXchange Project
<b>eX_Common_Environments.zip</b>	eXchange Environments
<b>4010_270_271_feeders.zip</b>	Input Project for 270/271 requests (optional for running the sample)
<b>4010_850_855_feeders.zip</b>	Input Project for 850/855 requests
<b>X12_Host.zip</b>	ASC X12 Logical Host Project
<b>HIPAA_270_271_feeders.zip</b>	Input Project for 270/271 HIPAA transactions (optional for running the sample)
<b>HIPAA_278a1_278a3_feeders.zip</b>	Input Project for 278 A1/178 A3 HIPAA transactions (optional for running the sample)

## 6.4 Importing the Sample Projects

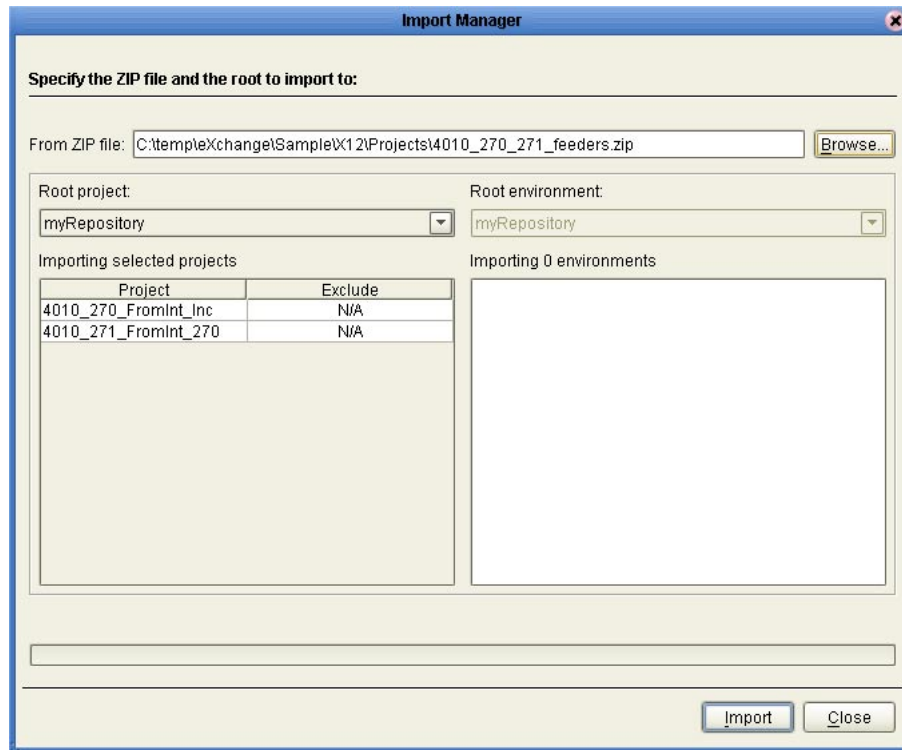
After unzipping the sample .zip file as described in the previous section, you can import the ASC X12 Manager sample into Enterprise Designer. The procedure below describes how you import the Project .zip files included in the X12 sample.

*Note: This procedure does not describe importing the HIPAA and 4010\_270\_271\_feeders.zip or HIPAA\_278a1\_278a3\_feeders.zip. Those Projects are optional.*

### 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 20** 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 ASC X12 Manager Composite Application” on page 30](#).
- 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\X12\Projects` folder.
- 9 Repeat step 5 and 6 to import the following Projects:
  - ♦ `4010_850_855_feeders.zip`
  - ♦ `X12Host.zip`
- 10 Click **Close**.

---

## 6.5 Understanding the 850 Feeder Project

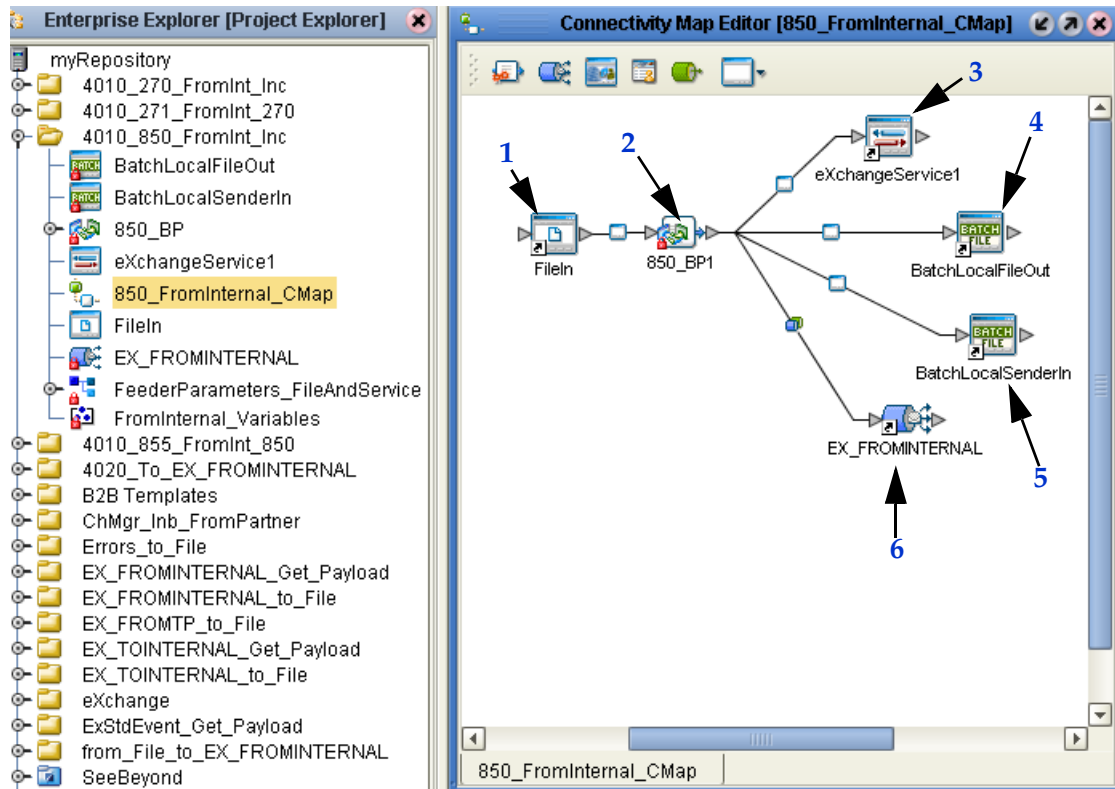
This section describes the Connectivity Map and BPs for the 850 feeder Project.



## 6.5.1 About the 850 Project Connectivity Map

The request feeder Project, **4010\_850\_FromInt\_Inc**, has a Connectivity Map named **850\_FromInternal\_CMap**. This is illustrated below and explained in the callouts that follow Figure 21.

**Figure 21** Connectivity Map for 850 Feeder Project



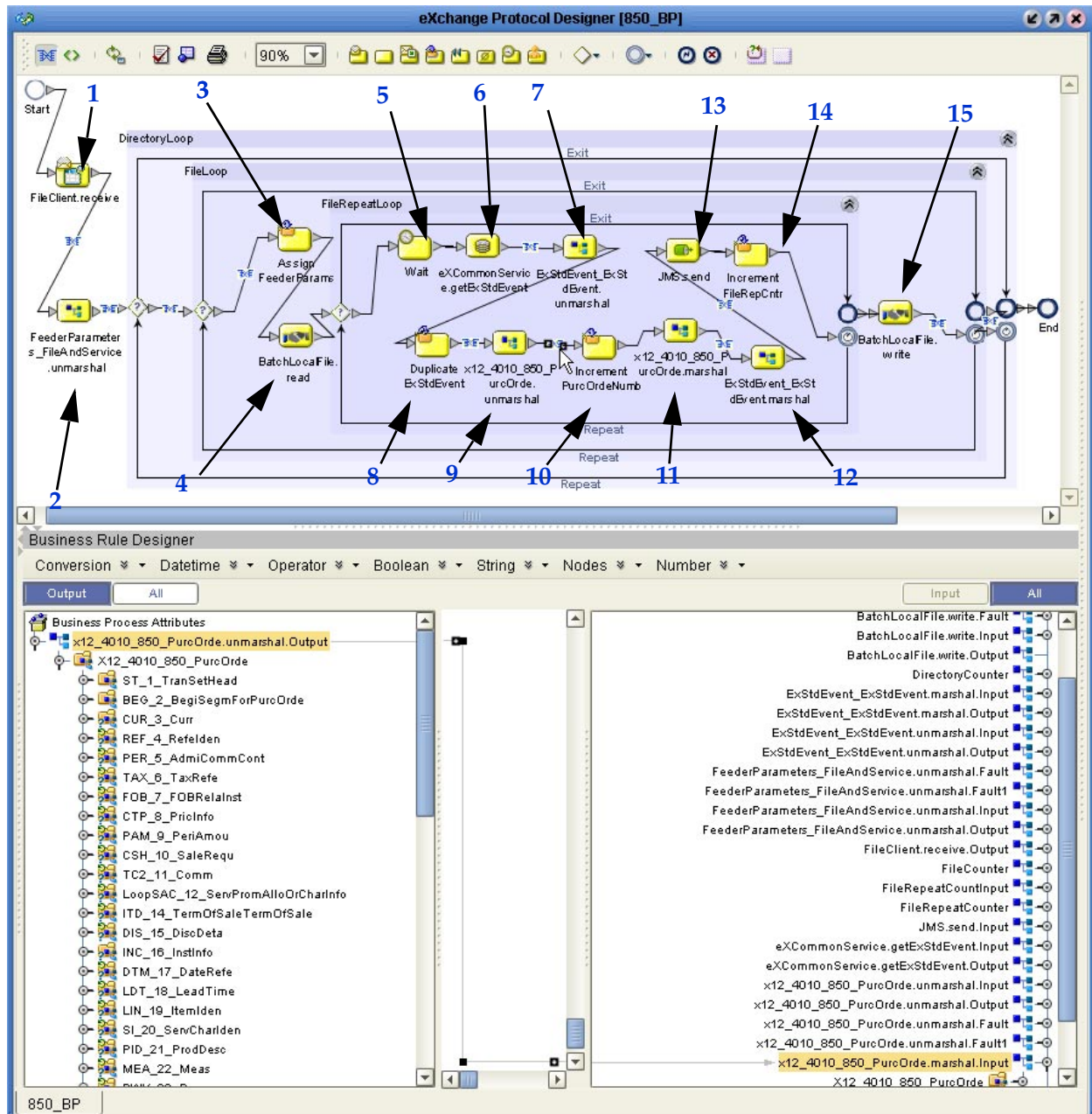
- 1 **FileIn eWay**: Connects to **850\_BP1** and supplies the TP, service, action, and direction information required to look up a Trading Partner, as well as a data source for the initial ExStdEvent payload.
- 2 **850\_BP1 Business Process**: Receives the data, looks up the trading partner, gets ExStdEvent, increments the ID, updates ExStdEvent, and publishes it to the **EX\_FROMINTERNAL** topic. For more information, refer to [“About the 850 Project BP” on page 57](#).
- 3 **eXchangeService1**: Provides services for the Atlanta host used by **850\_BP1** for retrieving the ExStdEvent message and looking up the Trading Partner.
- 4 **BatchLocalFileOut eWay**: Writes out the 850 file for Berlin.
- 5 **BatchLocalSenderIn eWay**: Reads the 850 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 850 Project BP

The 850\_BP1 initiates the data flow for the 850 feeder Project. Figure 22 shows the 850\_BP1 in the Enterprise Designer.

Figure 22 850\_BP1 for the 850 Feeder Project



850\_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 X12\_4010\_850\_PurcOrde” OTD, from **SeeBeyond** > **OTD Library** > **X12** > **v4010**): Unmarshals the template data that was read in.
- 10 *Increment PurcOrdeNumb business rule*: Increments the purchase order number.
- 11 *marshal Activity* (of the X12\_4010\_850\_PurcOrde” OTD, from **SeeBeyond** > **OTD Library** > **X12** > **v4010**): 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 850 out for the Trading Partner to retrieve.

---

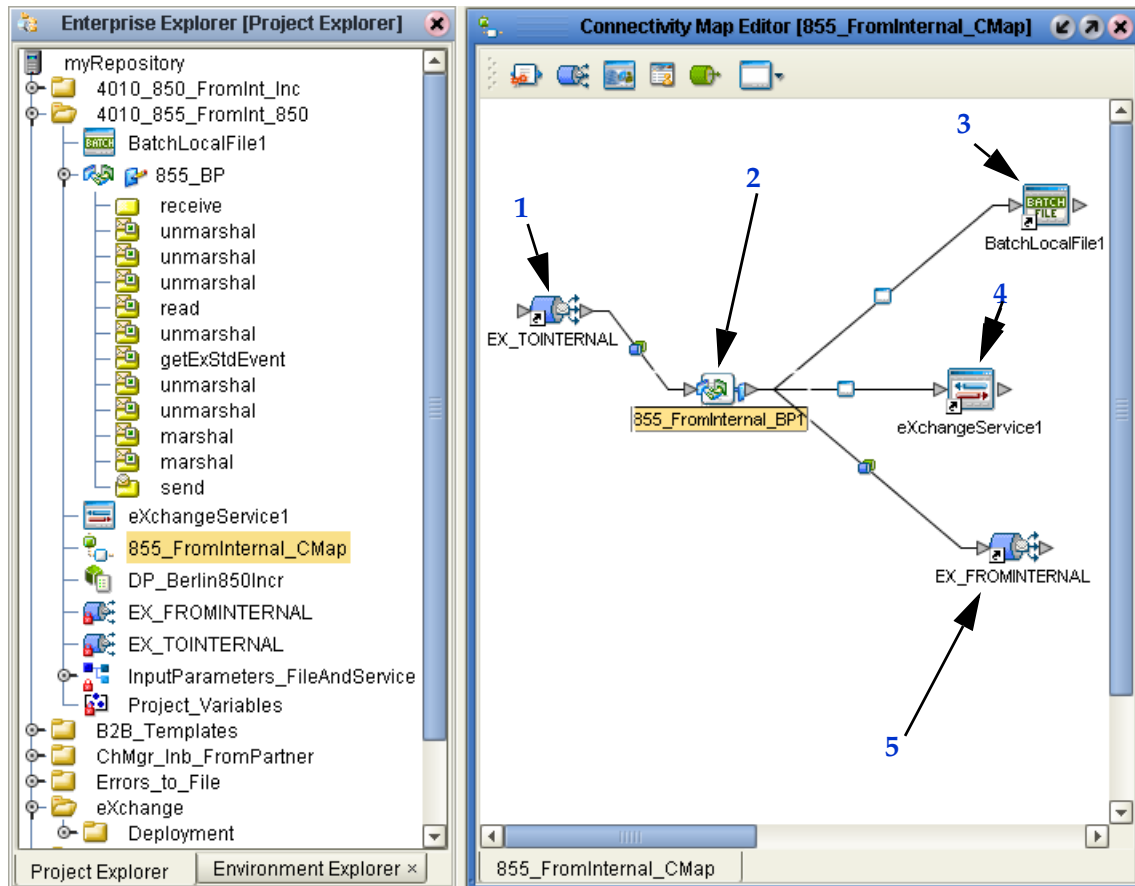
## 6.6 Understanding the 855 Feeder Project

This section describes the Connectivity Map and BPs for the 855 feeder Project.

### 6.6.1 About the 855 Project Connectivity Map

The response feeder Project, 4010\_855\_FromInt\_850, has a Connectivity Map named **855\_FromInternal\_CMap**. This is illustrated below and explained in the callouts that follow Figure 23.

**Figure 23** Connectivity Map for 855 Feeder Project

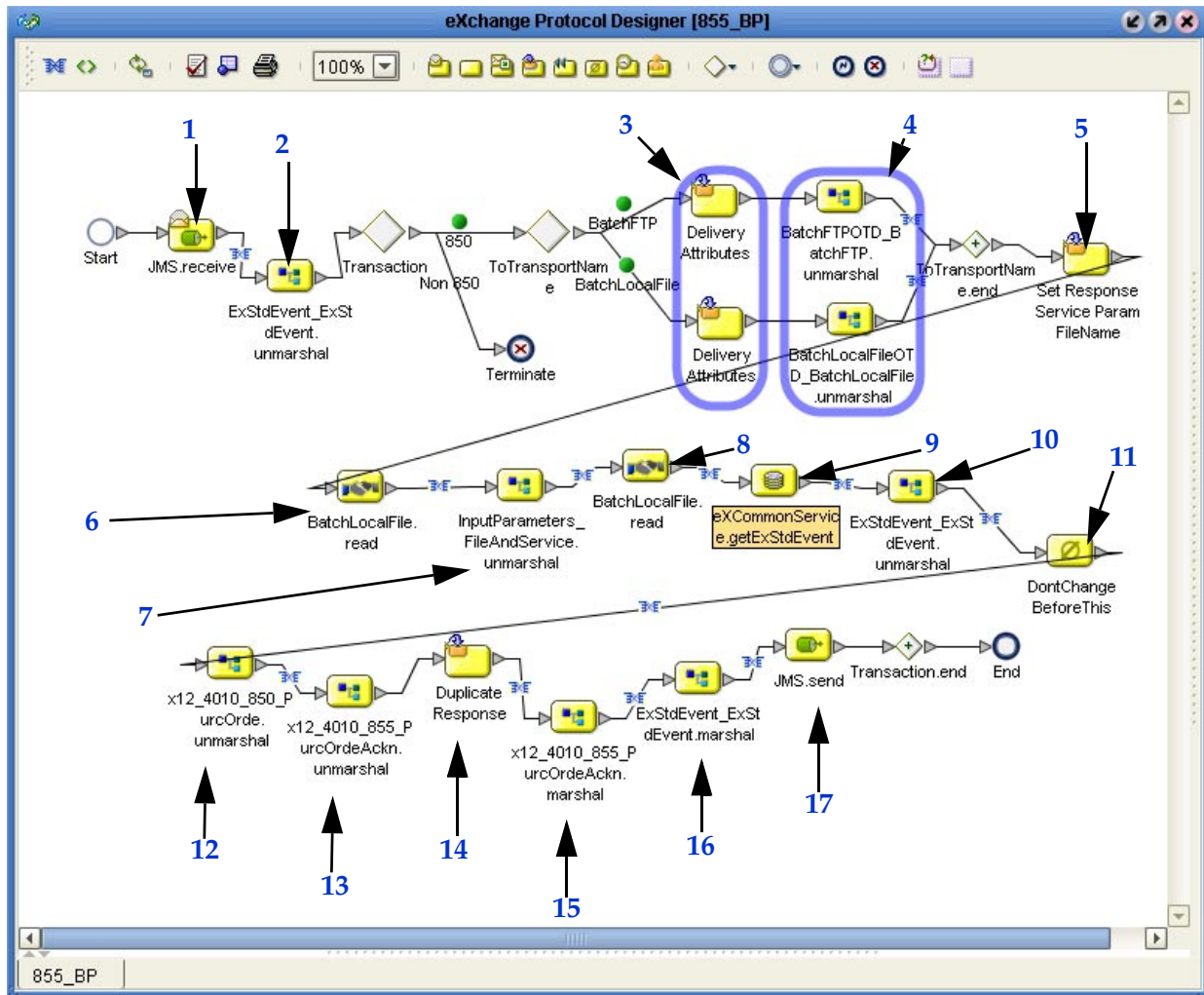


- 1 *EX\_TOINTERNAL* topic: A JMS topic whose publisher is the **bpEX\_SendToInternalAndDialogX12BP**, which provides the message from the trading partner, and whose subscriber is ICAN.
- 2 *855\_BP1*: Receives the data, looks up the Trading Partner, retrieves **ExStdEvent**, increments the ID, updates **ExStdEvent**, and publishes it to the **EX\_FROMINTERNAL** topic. For more information, refer to “[About the 855 Project BP](#)” on page 59.
- 3 *BatchLocalFile eWay*: Writes out the 855template file.
- 4 *eXchangeService1*: Provides services for the Berlin B2B host used by *855\_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 855 Project BP

The *855\_BP1* replies to a request for the 850 feeder Project. Figure 24 shows the *855\_BP1* in the Enterprise Designer.

Figure 24 855\_BP1 for the 855 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 855 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 855 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 X12\_4010\_850\_PurcOrde OTD, from SeeBeyond > OTD Library > X12 > v4010): Unmarshals the 850 message.
- 13 *unmarshal Activity* (of the X12\_4010\_855\_PurcOrde OTD, from SeeBeyond > OTD Library > X12 > v4010): Generates the 855 message.
- 14 *Duplicate Response business rule*: Verifies whether there are duplicates.
- 15 *marshal Activity* (of the X12\_4010\_855\_PurcOrde OTD, from SeeBeyond > OTD Library > X12 > v4010): Marshals the 855 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 32, 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	<i>SID for the eXchange database</i>
DataSourceName	<b>local</b>
Password	<b>ex_A</b>
PortNumber	<b>1521</b> (unless the Oracle administrator changed the default)



Option	Setting
ServerName	<i>name of the Oracle server</i>
User	<b>ex_A</b>

- 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**.

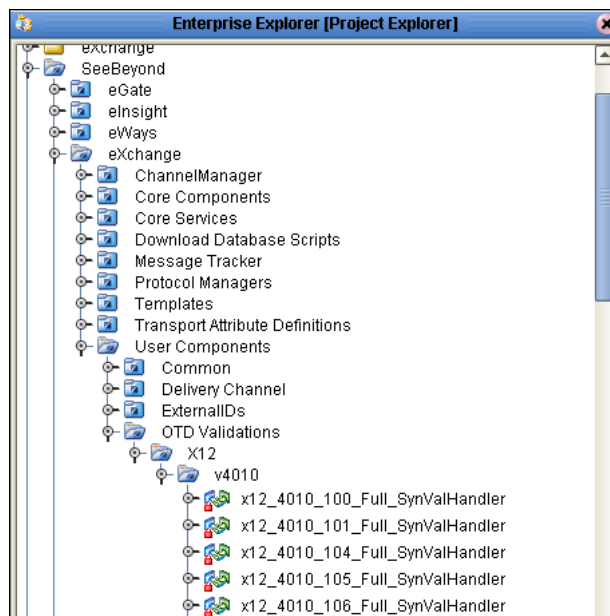
## 6.8 Creating the Validation Connectivity Map

The procedure below describes how to create a Connectivity Map to specify which validation BPs are to be used in the sample Project.

To create the validation Connectivity Map

- 1 In the **Project Explorer** tab in the Enterprise Designer, expand **eXchange**, right-click **Deployment**, click **New**, and click **Connectivity Map**.
- 2 Drag **eXchangeService** to the new map.
- 3 Expand the **eXchange**, **User Components**, **OTD Validations**, **X12**, and **v4010** folders.

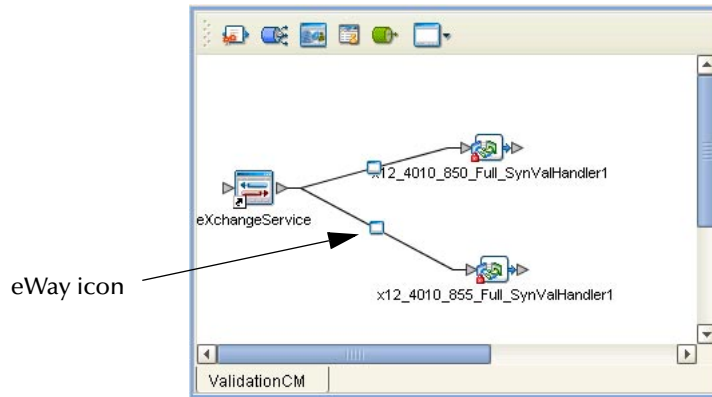
**Figure 25** Locating Validation BPs



- 4 Drag **X12\_4010\_850\_Full\_SynValHandler** and **X12\_4010\_855\_Full\_SynValHandler** onto the Connectivity Map.

- 5 Link **eXchangeService** to the 850 validation handler and to the 855 validation handler as shown below.

**Figure 26** Creating the Validation Connectivity Map



- 6 Double-click the eWay icon for each link, click **Inbound Exchange Service** in the **Templates** dialog box appears and click **OK**. The **Properties** dialog box appears.
- 7 Click **OK** and click **Save**. The Connectivity Map is now complete.

## 6.9 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 12** Deployment Profiles To Be Created and Activated

Project Name	Deployment Profile(s)	Extra Steps
X12_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 X12_Host eXchange Service for each profile.
Errors_to_File	2 profiles; one for AtlantaEnv and a second for BerlinEnv	Map to the <b>Directory</b> constant in each Environment before activating profiles.
4010_850_FromInt_Inc	1 profile for AtlantaEnv	Map to the <b>X12_DataDir</b> constant in the Environment before activating the profile.
4010_855_FromInt_850	1 profile for BerlinEnv	Map to the <b>X12_DataDir</b> constant in the Environment before activating the profile.

**Table 12** Deployment Profiles To Be Created and Activated

Project Name	Deployment Profile(s)	Extra Steps
[eXchange] Deployment	2 profiles; one for AtlantaEnv and a second for BerlinEnv	If you use the optional 270/271 Projects in addition to the 850/855 Project, create and activate the 270/271 profiles (only—do not activate the 850/855 profiles)
HIPAA_270_FromInt_Inc	1 profile for AtlantaEnv	<i>(optional— not necessary to run the sample)</i>
HIPAA_271_FromInt_850	1 profile for BerlinEnv	<i>(optional— not necessary to run the sample)</i>

**Note:** If the error “Failed to obtain JNDI name prefix(ERROR\_GET\_JNDINAMEPREFIX)” occurs, you forgot to map to the **X12\_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 **X12\_DataDir**, and click **OK**.
  - ♦ **Errors\_to\_File**
  - ♦ **4010\_850\_FromInt\_Inc**
  - ♦ **4010\_855\_FromInt\_850**
- 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 X12\_Host eXchange Service; select protocol **mad\_X12**.
- 4 Click **Save All**.
- 5 Click **Activate**. Click **No** when the Apply to Logical Host message appears.

---

## 6.10 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 **X12\_Host1**.
- 3 Enter **Atlanta** for the name, browse to:  
**C:\temp\exchange\Sample\TradingPartners\X12TP\_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\X12TP\_for-AtlantaEnv\_to-from\_Berlin.xml**.
- 5 Select Atlanta or Berlin, click **Activate**.
- 6 Click **Activate**.

---

## 6.11 Running the X12 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 66](#).

### 6.11.1 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.

## 6.11.2 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\X12\Data\Atlanta` folder, change the file extension `.~in` to `.in` for the following file:

`X12dlg_850_Out_feeder_Berlin.~in`

The Atlanta host ...

- Reads files of this form:  
`...\Sample\X12\Data\Atlanta\X12dlg_<action>_Out_feeder_Berlin.in`

The `.in` file is an XML file that points to the `C:\temp\exChange\Sample\Data\X12\Atlanta` folder and `_incremented.st` file.

- Writes output resulting from successful processing to files of this form:  
`...\Sample\X12\Data\Berlin\Atlanta_850_In_ISA_<timestamp>.dat`
- Writes error messages to files of this form:  
`...\Sample\X12\Data\Atlanta\ProcessedError_%d.dat`  
`...\Sample\X12\Data\Atlanta\DeadLetter_%d.dat`

The Berlin host ...

- Reads files of this form:  
`...\Sample\X12\Data\Berlin\X12*`
- Writes output resulting from successful processing to files of this form:  
`...\Sample\X12\Data\Atlanta\Berlin_850_Out_ISA_<timestamp>.dat`
- Writes error messages to files of this form:  
`...\Sample\X12\Data\Berlin\ProcessedError_%d.dat`  
`...\Sample\X12\Data\Berlin\DeadLetter_%d.dat`

# OTD Syntax Validation BPs

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

## What's in This Appendix

- **Activity Flow** on page 67
- **Fault Handling** on page 70
- **Variables Referenced by OTD Validation BPs** on page 72

---

## 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 70](#).

- 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 `BusinessTransactionType` 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 BusinessTransactionType 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 70](#).
- ◆ In the case of a *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

- [ValidateException](#) on page 70
- [UnmarshalException](#) on page 71
- [GenericException](#) on page 71
- [Other Faults](#) on page 71

### 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 13** Variables Referenced by OTD Validation BPs

Variable Name	Assignment (using X12 v4010 850 as an example)
\${OtdName}	X12_4010_850_PurcOrde_Full
\${HandlerName}	X12_4010_850_Full_SynValHandler
\${OuterNodeName}	X12_4010_850_PurcOrde_Outer
\${InnerNodeName}	X12_4010_850_PurcOrde_Inner
\${TransNodeName}	X12_4010_850_PurcOrde
\${TSHeaderNodeName}	ST_1_TransSetHead
\${BusinessTransactionIdentifier}	E143_1_TransSetIdenCode
\${TransactionExternalID}	E329_2_TransSetContNumb
\${FGHeaderNodeName}	GS_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\_RefIden
- X12\_004010X092A1\_00\_hipaa\_271\_EligCoveOrBeneInfo\_Full=BHT\_msk1\_2\_BegiOfHierTran/E127\_3\_RefIden

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\_RefIden
- X12\_004010X092\_00\_hipaa\_271\_EligCoveOrBeneInfo\_Full=BHT\_msk1\_2\_BegiOfHierTran/E127\_3\_RefIden

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

## A

Accredited Standards Committee 14  
 acknowledgments  
   application 26  
   as part of EDI logic 27  
   functional acknowledgment (997) 26  
   interchange acknowledgment (TA1) 26  
   receipt of payment order 26  
   types of 26  
 American National Standards Institute 14  
 ANSI 14  
 application acknowledgments 26  
 ASC 14

## B

backward compatibility 22

## C

configuring  
   Oracle eWay 61  
 control numbers 22  
   functional group control number (GS06) 22  
   interchange control number (ISA13) 22  
   transaction set control number (ST02) 22  
 conventions, document 10

## D

data element separator 17  
 data elements 16  
 Data Interchange Standards Association 14  
 delimiters 15, 17  
   data element separator 17  
   repetition separator 17  
   segment terminator 17  
   subelement (component) separator 17  
 DISA 14  
 document conventions 10

## E

EDI 14

  payment processing overview 23  
   usage example 23  
 EDISIM 27  
 enveloping  
   as part of EDI logic 27  
 example of EDI usage 23  
 eXchange support  
   for platforms 29

## F

finding sample Projects 53  
 Foresight Corporation 27  
 functional acknowledgments (997) 26  
 functional group 20  
 functional group control number (GS06) 22

## G

GS06 (functional group control number) 22

## H

heap size  
   adjusting heap memory size 32

## I

IC (interchange envelope) 21  
 implementation 27  
 importing sample Projects 54  
 interchange acknowledgment (TA1) 26  
 interchange control number (ISA13) 22  
 interchange envelope 21  
 ISA13 (interchange control number) 22

## L

loops 17

## M

message structure  
   defined 15  
   OTD in eGate 15

## O

operating systems supported by eXchange 29  
 Options Setup  
   dialog box 32  
 Oracle eWay, configuring 61  
 OutOfMemoryError

## Index

- increase heap size 32
- overview
  - of EDI payments processing 23
  - of X12 14–28
  - sample Projects 48

## P

- payment-related EDI transactions 25
- platforms supported by eXchange 29

## R

- repetition separator 17
- response transactions 26

## S

- sample Projects
  - finding 53
  - importing 54
  - overview 48
- Screenshots 10
- SEF file 27
- SEF OTD Wizard 27
- SEF OTD wizard
  - installing 31
- segment terminator 17
- segments 16
- ST02 (transaction set control number) 22
- structure of an X12 envelope 18
- structures 27
  - as part of EDI logic 27
- subelement (component) separator 17
- supporting documents 10
- syntax
  - control numbers 22
  - delimiters 17

## T

- TA1 (interchange acknowledgment) 26
- trading partner agreements 27
- transaction set control number (ST02) 22
- translations
  - as part of EDI logic 27

## U

- UN/EDIFACT standard 14

## V

- validations
  - as part of EDI logic 27

## W

- what is a message structure? 15

## X

- X12
  - acknowledgment types 26
  - data elements 16
  - end-to-end example 25
  - envelope structure 18
  - functional group 20
  - interchange envelope 21
  - loops 17
  - segments 16
    - what is it? 14
- X12 body 14
- X12 overview 14–28