SeeBeyond™ eBusiness Integration Suite

UN/EDIFACT ETD Library User's Guide

Release 4.5.2

SeeBeyond Proprietary and Confidential

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.

e*Gate, e*Insight, e*Way, e*Xchange, e*Xpressway, eBI, iBridge, Intelligent Bridge, IQ, SeeBeyond, and the SeeBeyond logo are trademarks and service marks of SeeBeyond Technology Corporation. All other brands or product names are trademarks of their respective companies.

© 1999–2002 by SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

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

Version 20020222105538.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

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

Note: *If you are upgrading to version 4.5.2 from a previous version of this library, you must recompile all Collaborations that use these ETDs after installing the new version.*

1.1 User's Guide Purpose and Scope

This document acquaints you with the SeeBeyond Technology Corporation™ (SeeBeyond™) UN/EDIFACT ETD Library, which is a feature of the SeeBeyond e*Gate Integrator system, and tells you how to install and use the UN/EDIFACT ETD Library.

This manual includes:

- Overview of UN/EDIFACT, including examples of a batch message and a segment table
- Procedures for installing UN/EDIFACT files and where to find them after installation
- Lists of sample file and directory names in the UN/EDIFACT ETD Library

Important: *Any operation explanations given here are generic, for reference purposes only, and do not necessarily address the specifics of setting up and/or operating individual e*Gate systems.*

This document does not contain information on software installation and system administration procedures (see **"Supporting Documents" on page 11**).

1.2 Intended Audience

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the e*Gate system, to have moderate to advanced level knowledge of Windows NT and Windows 2000 operations and administration, and to be thoroughly familiar with Windows-style GUI operations.

1.3 Document Organization

This document is organized topically as follows:

- **Chapter 1 "Introduction"** gives a general preview of this document, its purpose, scope, and organization.
- **Chapter 2 "Overview of UN/EDIFACT"** provides an overview of UN/EDIFACT, including examples of a batch message and a segment table, along with additional information about their components, structure, and validation rules.
- **Chapter 3 "UN/EDIFACT Template Installation"** explains how to install UN/EDIFACT files and where to find them after installation.
- **Chapter 4 "UN/EDIFACT ETD Library"** lists sample file and directory names in the UN/EDIFACT ETD Library.

1.4 Writing Conventions

The writing conventions listed in this section are observed throughout this document.

Hypertext Links

When you are using this guide online, cross-references are also hypertext links and appear in **blue text** as shown below. Click the **blue text** to jump to the section.

> For information on these and related topics, see **"Parameter, Function, and Command Names" on page 10**.

Command Line

Text to be typed at the command line is displayed in a special font as shown below.

```
java -jar ValidationBuilder.jar
```

Variables within a command line are set in the same font and bold italic as shown below.

```
stcregutil -rh host-name -rs schema-name -un user-name
-up password -ef output-directory
```

Code and Samples

Computer code and samples (including printouts) on a separate line or lines are set in Courier as shown below.

```
Configuration for BOB_Promotion
```

However, when these elements (or portions of them) or variables representing several possible elements appear within ordinary text, they are set in *italics* as shown below.

> *path* and *file-name* are the path and file name specified as arguments to **-fr** in the **stcregutil** command line.

Notes and Cautions

Points of particular interest or significance to the reader are introduced with ***Note***, ***Caution***, or ***Important***, and the text is displayed in *italics*, for example:

Note: *The Actions menu is only available when a Properties window is displayed.*

User Input

The names of items in the user interface such as icons or buttons that you click or select appear in **bold** as shown below.

Click **Apply** to save, or **OK** to save and close.

File Names and Paths

When names of files are given in the text, they appear in **bold** as shown below.

Use a text editor to open the **ValidationBuilder.properties** file.

When file paths and drive designations are used, with or without the file name, they appear in **bold** as shown below.

In the **Open** field, type **D:\setup\setup.exe** where **D:** is your CD-ROM drive.

Parameter, Function, and Command Names

When names of parameters, functions, and commands are given in the body of the text, they appear in **bold** as follows:

The default parameter **localhost** is normally only used for testing.

The Monk function **iq-put** places an Event into an IQ.

You can use the **stccb** utility to start the Control Broker.

Additional Conventions

Windows Systems — The e*Gate system is fully compliant with both Windows NT and Windows 2000 platforms. When this document references Windows, such statements apply to both Windows platforms.

UNIX Systems — This guide uses the backslash ("\") as the separator within path names. If you are working on a UNIX system, please make the appropriate substitutions.

Note: *e*Gate is fully compatible with Compaq Tru64 V4.0F and V5.0A.*

1.5 Supporting Documents

The following SeeBeyond documents provide additional information about the Alerts and log files as explained in this guide:

- *ASC X12 ETD Library User's Guide*
- *Batch e*Way Intelligent Adapter User's Guide*
- *e*Gate Integrator Alert Agent User's Guide*
- *e*Gate Integrator Installation Guide*
- *e*Gate Integrator Intelligent Queue Services Reference Guide*
- *e*Gate Integrator SNMP Agent User's Guide*
- *e*Gate Integrator System Administration and Operations Guide*
- *e*Gate Integrator User's Guide*
- *Monk Developer's Reference*
- *SeeBeyond eBusiness Integration Suite Deployment Guide*
- *SeeBeyond eBusiness Integration Suite™ Primer*
- *Standard e*Way Intelligent Adapters User's Guide*

See the *e*Insight eBusiness Integration Suite Primer* for a complete list of e*Gate-related documentation. You can also refer to the appropriate Microsoft Windows or UNIX documents, if necessary.

1.6 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

Chapter 2

Overview of UN/EDIFACT

This chapter presents an overview of UN/EDIFACT, including examples of a batch message and a segment table, along with additional information about their components, structure, and validation rules.

2.1 UN/EDIFACT Components

UN/EDIFACT stands for United Nations/Electronic Data Interchange for Administration, Commerce and Transport. It is a standard, developed for the electronic exchange of machine-readable information between businesses.

The UN/EDIFACT Working Group (EWG) develops, maintains, interprets, and promotes the proper use of the UN/EDIFACT standard. UN/EDIFACT is broadly used in Europe and other parts of the world.

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

The UN/EDIFACT ETD Library allows e*Gate and e*Xchange customers to easily visualize the structures within a graphical user interface and to build up business rules (Collaborations) through drag and drop technology.

Monk ETD files

Monk ETD (**.ssc**) files define the structure and syntax of message formats that are used to identify, validate, and translate message data content. An ETD library includes ETD files for industry standard message formats. The UN/EDIFACT ETD Library includes ETD files for all messages in UN/EDIFACT versions D95A through D01B.

In the UN/EDIFACT ETD Library, there is a separate subdirectory for each Monk version of UN/EDIFACT, for example: **edifact_d99a**, **edifact_d99b**. Each version directory contains all transaction sets for that version (for more information, see **"ETD Libraries" on page 22**). ETD files for segments contained in the transaction sets are in their own subdirectory, so the user can select from only the transaction sets, unless further drill-down is desired.

Java ETD files

Java ETD (**.xsc**) files define the structure and syntax of message formats that are used to identify, validate, and translate message data content. Java ETDs also contain **.jar** files, which function much like Zip files in that they compress and store Java **.class** files. The **.class** files support the **.xsc** structure (message, parsing, and validation). Java also uses a Standard Exchange Format (SEF) file, which allows users to add extra validation scripts.

Since Java ETDs function differently than the Monk ETDs, a separate UN/EDIFACT ETD Library, with separate subdirectories for each Java version of UN/EDIFACT must also be maintained, for example: **java_edifact_d99a**, **java_edifact_d99b**. Each version directory contains all transaction sets for that version.

Monk UN/EDIFACT messages

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

The messages are published on the Web at:

http://www.itsedi.com/henry/

The envelopes are published on the Web at:

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

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

Java UN/EDIFACT messages

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

Java uses a secondary UN/EDIFACT format that is different from the standard UN/EDIFACT format. The secondary format uses a SEF file, which has structure, as well as methods and functions that can act upon the message.

2.1.1 Message Structure

The term *message structure* (also called a transaction set structure) refers to the way in which data elements are organized and related to each other for a particular EDI transaction.

In e*Gate, a message structure is called an Event Type Definition (ETD). Each message structure (ETD) consists of the following:

- Physical hierarchy

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

- Delimiters

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

- Properties

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

For example, the transaction set structure of invoices that you send to trading partners defines the header, trailer, segment, and data element required by invoice transactions. Installation of UN/EDIFACT ETD Library templates 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.

e*Gate uses the message structure to interpret the actual message coming in or going out. There is a message structure for each UN/EDIFACT transaction.

The list of transactions provided is different for each version of UN/EDIFACT.

2.1.2 Messages

As an example, Table 1 below lists the batch messages, along with their function, that are defined in version D00A:

Table 1 Batch Messages Defined in Version D00A

Name	Function
APERAK	Application error and acknowledgement message
AUTHOR	Authorization message
BALANC	Balance message
BANSTA	Banking status message
BAPLIE	Bayplan/stowage plan occupied and empty locations message
BAPLTE	Bayplan/stowage plan total numbers message
BERMAN	Berth management message
BMISRM	Bulk marine inspection summary report message
BOPBNK	Bank transactions and portfolio transactions report message
BOPCUS	Balance of payment customer transaction report message
BOPDIR	Direct balance of payment declaration message
BOPINF	Balance of payment information from customer message
BUSCRD	Business credit report message
CALINF	Vessel call information message
CASINT	Request for legal administration action in civil proceedings message
CASRES	Legal administration response in civil proceedings message

Table 1 Batch Messages Defined in Version D00A (Continued)

Name	Function
CHACCO	Chart of accounts message
CLASET	Classification information set message
CNTCND	Contractual conditions message
COACSU	Commercial account summary message
COARRI	Container discharge/loading report message
CODECO	Container gate-in/gate-out report message
CODENO	Permit expiration/clearance ready notice message 5
COEDOR	Container stock report message
COHAOR	Container special handling order message
COLREQ	Request for a documentary collection message
COMDIS	Commercial dispute message
CONAPW	Advice on pending works message
CONDPV	Direct payment valuation message
CONDRA	Drawing administration message
CONDRO	Drawing organization message
CONEST	Establishment of contract message
CONITT	Invitation to tender message
CONPVA	Payment valuation message
CONQVA	Quantity valuation message
CONRPW	Response of pending works message
CONTEN	Tender message
CONWQD	Work item quantity determination message
COPAYM	Contributions for payment
COPARN	Container announcement message
COPINO	Container pre-notification message
COPRAR	Container discharge/loading order message
COREOR	Container release order message
COSTCO	Container stuffing/stripping confirmation message
COSTOR	Container stuffing/stripping order message
CREADV	Credit advice message
CREEXT	Extended credit advice message
CREMUL	Multiple credit advice message
CUSCAR	Customs cargo report message
CUSDEC	Customs declaration message
CUSEXP	Customs express consignment declaration message

Table 1 Batch Messages Defined in Version D00A (Continued)

Name	Function
CUSPED	Periodic customs declaration message
CUSREP	Customs conveyance report message
CUSRES	Customs response message
DEBADV	Debit advice message
DEBMUL	Multiple debit advice message
DEBREC	Debts recovery message
DELFOR	Delivery schedule message
DELJIT	Delivery just in time message
DESADV	Despatch advice message
DESTIM	Equipment damage and repair estimate message
DGRECA	Dangerous goods recapitulation message
DIRDEB	Direct debit message
DIRDEF	Directory definition message
DMRDEF	Data maintenance status report/query message
DMSTAT	Data maintenance status report/query message
DOCADV	Documentary credit advice message
DOCAMA	Advice of an amendment of a documentary credit message
DOCAMI	Documentary credit amendment information message
DOCAMR	Request for an amendment of a documentary credit message
DOCAPP	Documentary credit application message
DOCARE	Response to an amendment of a documentary credit message
DOCINF	Documentary credit issuance information message
ENTREC	Accounting entries message
FINCAN	Financial cancellation message
FINPAY	Multiple interbank funds transfer message
FINSTA	Financial statement of an account message
GENRAL	General purpose message
GESMES	Generic statistical message
HANMOV	Cargo/goods handling and movement message
IFCSUM	Forwarding and consolidation summary message
IFTCCA	Forwarding and transport shipment charge calculation message
IFTDGN	Dangerous goods notification message
IFTFCC	International transport freight costs and other charges message

Table 1 Batch Messages Defined in Version D00A (Continued)

Name	Function
IFTIAG	Dangerous cargo list message
IFTMAN	Arrival notice message
IFTMBC	Booking confirmation message
IFTMBF	Firm booking message
IFTMBP	Provisional booking message
IFTMCA	Consignment advice message
IFTMCS	Instruction contract status message
IFTMIN	Instruction message
IFTRIN	Forwarding and transport rate information message
IFTSAI	Forwarding and transport schedule and availability information message
IFTSTA	International multimodal status report message
IFTSTQ	International multimodal status request message
IMPDEF	EDI implementation guide definition message
INFENT	Enterprise accounting information message
INSDES	Instruction to despatch message
INSPRE	Insurance premium message
INSREQ	Inspection request message
INSRPT	Inspection report message
INVOIC	Invoice message
INVRPT	Inventory report message
IPPOAD	Insurance policy administration message
IPPOMO	Motor insurance policy message
ITRRPT	In transit report detail message
JAPRES	Job application result message
JINFDE	Job information demand message
JOBAPP	Job application proposal message
JOBCON	Job order confirmation message
JOBMOD	Job order modification message
JOBOFF	Job order message
JUPREQ	Justified payment request message
LEDGER	Ledger message
LREACT	Life reinsurance activity message
LRECLM	Life reinsurance claims message
MEDPID	Person identification message
MEDPRE	Medical prescription message

Table 1 Batch Messages Defined in Version D00A (Continued)

Name	Function
MEDREQ	Medical service request message
MEDRPT	Medical service report message
MEDRUC	Medical resource usage and cost message
MEQPOS	Means of transport and equipment position message
MOVINS	Stowage instruction message
MSCONS	Metered services consumption report message
ORDCHG	Purchase order change request message
ORDERS	Purchase order message
ORDRSP	Purchase order response message
OSTENQ	Order status enquiry message
OSTRPT	Order status report message
PARTIN	Party information message
PAXLST	Passenger list message
PAYDUC	Payroll deductions advice message
PAYEXT	Extended payment order message
PAYMUL	Multiple payment order message
PAYORD	Payment order message
PRICAT	Price/sales catalogue message
PRIHIS	Pricing history message
PROCST	Project cost reporting message
PRODAT	Product data message
PRODEX	Product exchange reconciliation message
PROINQ	Product inquiry message
PROSRV	Product service message
PROTAP	Project tasks planning message
PRPAID	Insurance premium payment message
QALITY	Quality data message
QUOTES	Quote message
RDRMES	Raw data reporting message
REBORD	Reinsurance bordereau message
RECADV	Receiving advice message
RECALC	Reinsurance calculation message
RECECO	Credit risk cover message
RECLAM	Reinsurance claims message
RECORD	Reinsurance core data message

Table 1 Batch Messages Defined in Version D00A (Continued)

Name	Function
REGENT	Registration of enterprise message
RELIST	Reinsured objects list message
REMADV	Remittance advice message
REPREM	Reinsurance premium message
REQDOC	Request for document message
REQOTE	Request for quote message
RESETT	Reinsurance settlement message
RESMSG	Reservation message
RETACC	Reinsurance technical account message
RETANN	Announcement for returns message
RETINS	Instruction for returns message
RPCALL	Repair call message
SAFHAZ	Safety and hazard data message
SANCRT	International movement of goods governmental regulatory message
SLSFCT	Sales forecast message
SLSRPT	Sales data report message
SOCADE	Social administration message
SSIMOD	Modification of identity details message
SSRECH	Worker's insurance history message
SSREGW	Notification of registration of a worker message 1
STATAC	Statement of account message
STLRPT	Settlement transaction reporting message
SUPCOT	Superannuation contributions advice message
SUPMAN	Superannuation maintenance message
SUPRES	Supplier response message
TANSTA	Tank status report message
TAXCON	Tax control message
TPFREP	Terminal performance message
VATDEC	Value added tax message
VESDEP	Vessel departure message
WASDIS	Waste disposal information message
WKGRDC	Work grant decision message
WKGRRE	Work grant request message

2.1.3 Segment Table

A key section for each document appears the end of each message, section "4.3.1 Segment Table." Here you will find a segment table displaying the message structure which shows the order of segments and the manner in which they repeat. For example, below is the segment table for APERAK (a batch message; see **Table 1 on page 14**):

```
4.3.1 Segment table

Pos   Tag Name                                  S  R

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

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

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

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

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

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

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

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

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

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

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

2.1.4 Loops

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

- Locate the fields by specifying:
  - Transaction set (for example, APERAK)
  - Loop (for example, segment group 1)
  - Which occurrence of the loop
  - Segment (for example, DOC)
  - Field number (for example, DOC00)
  - Which occurrence of the segment (if repeating)

2.1.5 Envelopes

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

Note: *Interactive messages first appeared in the D96B message directory release.*

Table 2 X12-UN/EDIFACT Envelope Comparison

X12		ENVELOPE	UN/EDIFACT Batch Messages		UN/EDIFACT Interactive Messages	
start	end		start	end	start	end
ISA	IEA	Interchange Envelope	UNA/UNB	UNZ	UNA/UIB	UIZ
GS	GE	Functional Group Envelope	UNG	UNE	N/A	N/A
ST	SE	Message Envelope	UNH	UNT	UIH	UIT

UNA segment

All UN/EDIFACT message templates have a UNA segment, but these segments are optional and seldom used. The UNA segment can be found in the segment template in the 'template' subdirectory in case it is needed. It is used to send unusual delimiter characters.

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

The UNA segment template is a fixed length with segment ID = UNA, followed by 6 one-byte fields labelled "delimiter<n>."

Control messages

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

Each version of the UN/EDIFACT ETD Library includes both a v3 **CONTRL** and a v4 **CONTRL** message. The user can select which one to use.

2.1.6 Delimiters

Java

The delimiters in Java are dynamically set.

Monk

The structures in the Monk UN/EDIFACT ETD Library use the following default delimiters:

- Component = : (colon)
- Element = + (plus)
- Decimal = . (period)
- Release Mark = ? (question mark)
- Reserved for Future Use = * (asterisk)

Note: *The Reserved for Future Use delimiter is used for repetition.*

- Segment = ' (single quote)

In Monk, if you need different delimiters, open the message in the **ETD Editor** and choose the **File > Default Delimiters** menu option to change the delimiters.

2.1.7 ETD Libraries

The UN/EDIFACT ETD Library contains a separate sub-directory for each version of UN/EDIFACT, and within each version directory, all the segment templates are kept in a sub-directory. Because of this, the user only has to select from the messages, and not from the segments.

To search for Java files in directories:

<egate>/server/registry/repository/default/etd/templates/edifact/edifact_dnnn/v3

or

<egate>/server/registry/repository/default/etd/templates/edifact/edifact_dnnn/v4

Important: *The Java **.xsc** files are read only. You should not attempt to edit **.xsc** files.*

If you need to make changes to a Java file (**.xsc** or **.jar**), modify the SEF file, which is a text file, and regenerate. To Modify a SEF file:

1. With the Java ETD Editor open, select **File > New**.
2. From the **New Event Type Definition** window choose the **SEFWizard** and click **OK**.
3. Step through the SEFWizard until the **SEF Wizard - Step 1** dialog box appears.
   - A. Select A SEF file, by either using the **Browse** button to locate and existing file or entering a new name in the **SEF File Name** box.
   - B. In the **Optional Set Description File Name** box, use the **Browse** button to locate an existing description file or enter a new name in the box.
   - C. In the **Optional SEC Description File Name** box, use the **Browse** button to locate and existing SEC file or enter a new name in the box.
   - D. In the **Package Name** box, enter a package name for this SEF file; for example: **custominPackage**.
   - E. To make SEF files more compact, they do not have descriptions. If you need information added to the node names, use the **Use Descriptive Node Names** radio buttons to add a description to the node names. The default is **Yes**.
   - F. When satisfied with the information you have entered on this dialog box, click **Next**.
4. When the **SEF Wizard - Step 2** dialog box appears, review the wizard's summary. If the information is correct, click **Finish** to generate an Event Type Definition and its associated Java classes.

Note: *If the information is not correct, click **Back** to change your selections.*

To search for Monk files in directories:

<egate>/server/registry/repository/default/monk_scripts/templates/edifact/edifact_dnnn/v3

or

<egate>/server/registry/repository/default/monk_scripts/templates/edifact/edifact_dnnn/v4

To edit Monk library structures:

Open and "save as" to your schema, then cut unused segments.

2.1.8 UN/EDIFACT Versus X12

Since the 1960s, more and more industries use EDI. Although some have invented their own sets of standardized data formats, the following sets are the accepted standards:

- ASC X12 is used within the United States
- UN/EDIFACT is used across international industries

2.1.9 Security

EDI-INT is an international standard for secure EDI transmissions, both UN/EDIFACT and X12. It is emerging as a widespread EDI security standard. It has the following features:

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

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

For additional information:

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

2.2 Examples of EDI Usage

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

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

2.2.1 Overview of EDI Payments Processing

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

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

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

Exchange of remittance information

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

- Directly between trading partners or through their respective EDI VAN mailboxes
- Through the banking system, with the beneficiary receiving notice from his bank

- By the originator to the originator's bank as an order to pay, which in turn reports to the beneficiary

Routing of remittance information

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

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

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

Exchange of payment orders

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

- Credit transfer, initiated by the payor

  or

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

Regardless of how the credit transfer was initiated, the payor sends a payment order to its bank in one of two forms:

- X12 Payment Order/Remittance Advice (transaction set 820)
- UN/EDIFACT PAYEXT message

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

Functions a payment must perform

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

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

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

- Together through the banking system

  or

- Separated and traveling by different routes

Formats for transporting a payment

The X12 820 and the UN/EDIFACT PAYEXT are data formats for transporting a payment order from the originator to its bank. This payment order is either an:

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

  or

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

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

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

Issuance of a payment order

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

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

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

- Individual item approval
- Blanket approval of all incoming debits with an upper-dollar limit

- Blanket approval for a particular trading partner to originate any debit
- Some combination of the above

2.2.2 Payment-Related EDI Transactions

X12 and UN/EDIFACT route the Payment Order/Remittance Advice from the originator to the beneficiary in a different manner. X12 uses an end-to-end method whereas UN/EDIFACT uses a point-to-point method.

X12

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. The 820 is wrapped in an ACH banking transaction for the actual funds transfer between the banks. For an X12-UN/EDIFACT Payment Order/Remittance Advice comparison, see **Table 3 on page 27**. **Table 4 on page 28** lists other related X12-UN/EDIFACT transactions.

UN/EDIFACT

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

- The originator company uses the Payment Order to notify its bank that a funds transfer should take place (PAYEXT, PAYMUL).
- The originator company uses the Remittance Advice to notify the beneficiary of the payment (REMADV from originator, CREADV/DEBADV to beneficiary).
- The EFT actually moves the monetary value from one bank to another bank (ACH in the United States; SWIFT or CHIPS in Europe). For an X12-UN/EDIFACT Payment Order/Remittance Advice comparison, see Table 3 below. Table 4 lists other related X12-UN/EDIFACT transactions.

Table 3 X12-UN/EDIFACT Payment Order/Remittance Advice Comparison

Step	X12	Description of Action	UN/EDIFACT	Step
1	820	Payment Order from originator to its bank	PAYEXT, PAYMUL	1
1	820	Remittance Advice from originator to be passed on to beneficiary	REMADV	2
2	820	Remittance advice to beneficiary	CREADV, DEBADV	2
3	ACH containing 820	EFT between banks	SWIFT, CHIPS in Europe; ACH in the U.S.	3

Table 4 Other Related Transactions

X12	Transaction	UN/EDIFACT
828	Debit Authorization	AUTHOR
829	Payment Cancellation Request	FINCAN
831	Application Control Totals	(none)

2.2.3 Understanding Enveloping Scenarios

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

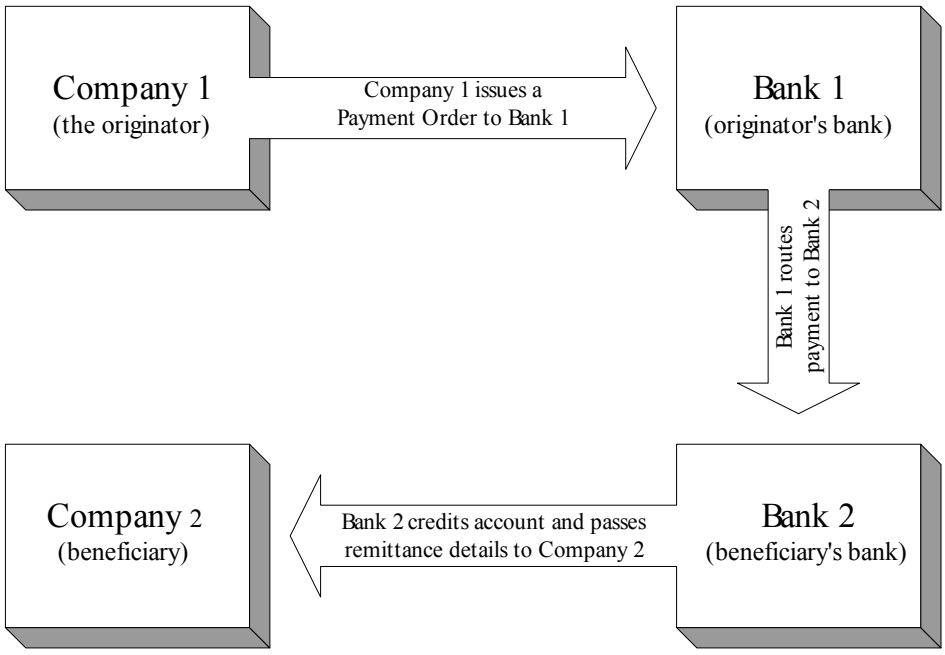
- Point-to-point
- End-to-end

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

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

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

Figure 1 Example Payment Scenario



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

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

In the 820 or PAYEXT, the identity of the originator, the originator's bank, the beneficiary's bank, and the beneficiary are established in the header of the transaction set (message) itself. Table 5, below, shows X12 and UN/EDIFACT headers.

Table 5 Sample X12 and UN/EDIFACT Headers

X12		Envelope	UN/EDIFACT	
start	**end**		**start**	**end**
ISA	IEA	Interchange Envelope	UNA/ UNB	UNZ
GS	GE	Functional Group Envelope	UNG	UNE
ST	SE	Message Envelope	UNH	UNT

Point-to-point scenario

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

1. Company 1 sends a payment file to Bank 1 using an X12 ISA (or UN/EDIFACT UNA/UNB) interchange header in which:
   - Sender ID = Company 1
   - Receiver ID = Bank 1
2. Bank 1 replaces Company 1's ISA (UNA/UNB) with its own ISA as follows:
   - Sender ID = Bank 1
   - Receiver ID = Bank 2
3. Bank 2 receives the payment file, and creates a new ISA to send the contents to Company 2 that shows:
   - Sender ID = Bank B
   - Receiver ID = Company B

Because the interchange control header (ISA or UNA/UNB) changes at each point, it is important that the functional group header (X12 GS or UN/EDIFACT UNG) is not changed.

Note: *The UN/EDIFACT functional group header (UNG) is optional.*

Maintaining the functional group guarantees that the payment file retains the Company 2 information. As some originators do not care what happens to the original ISA, it is imperative that each bank in the chain ensure that the X12 GS contains 820s (or PAYEXTs) that are destined for only one Company 2. This rule makes it so that the banks only have to look at the ISA for addressing information, and the receiving company can respond with a Functional Acknowledgment (X12 997 or UN/EDIFACT APERAK) to the originator.

End-to-end scenario

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

1. Company 1 sends a payment file to Bank 1 using an ISA in which:
   - Sender ID = Company 1
   - Receiver ID = Company 2
2. Bank 1 does not disturb the ISA, which continues to show:
   - Sender ID = Company 1
   - Receiver ID = Company 2
3. Bank 2 does not disturb the ISA, which continues to show:
   - Sender ID = Company 1
   - Receiver ID = Company 2

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

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

X12 recommends using the point-to-point addressing in the interchange header and end-to-end addressing in the functional group header.

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

2.2.4 Payment Acknowledgments

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

Note: *In UN/EDIFACT, CONTRL is a point-to-point acknowledgment.*

For examples of UN/EDIFACT and X12 acknowledgments, see Table 6.

Table 6 Types of UN/EDIFACT and X12 Acknowledgments

UN/EDIFACT	Envelope	X12
CONTRL	System Level Acknowledgment (receipt)	TA1
CONTRL	Function Acknowledgment (point-to-point)	997
	Application Advice (end-to-end)	824

2.3 Implementation

This section discusses implementation in e*Gate.

2.3.1 Implementation in e*Gate

The five key terms of EDI processing logic are listed in Table 7 below. In order to be clear about the meaning of terms, we can distinguish five parts of EDI processing logic:

Table 7 Key Terms of EDI Processing

Term	Description	Language Analogy	e*Gate Collaboration Scripts
Structures	Format, segments, loops	Syntax	ETD files or structures
Validations	Data contents “edit” rules	Semantics	Validation scripts
Translations	Re-formatting or conversion	Translation	Translation scripts
Enveloping	Header and trailer segments	Envelopes	Part of translation
Acks	Acknowledgments	Return receipt	e*Way scripts
Note: Java does not generate acknowledgments.			

The UN/EDIFACT ETD Library supplies UN/EDIFACT structures, that is, the first row in the above table. Other parts of the SeeBeyond product suite support the other functions.

Note: *Translations are also known as mappings.*

Structures

In e*Gate, all UN/EDIFACT transactions for all UN/EDIFACT versions are pre-built and part of the UN/EDIFACT structure library.

Monk customization

To customize the Monk structure, use the e*Gate Monk Event Type Definition Editor.

Note: *For more information on the Monk Event Type Definition Editor, see the* ***e*Gate Integrator User's Guide****.*

Java customization

To customize the Java structure (for example: adding a segment or loop), you must use third party tools, such as the EDISIM tool (Foresight Corporation), to generate a SEF file. You would then use the e*Gate Java Event Type Definition Editor and the SEFWizard to generate the **.xsc** and **.jar** files.

Note: *For more information on the Java Event Type Definition Editor and the SEFWizard, see the* ***e*Gate Integrator User's Guide****.*

Validations, translations, enveloping, and acknowledgments

e*Gate does not contain any pre-built Monk validations, translations, enveloping, or acknowledgments. These scripts can be built in the Monk Collaboration Rules Editor graphical user interface (GUI), which provides a user-friendly drag-and-drop front end for creating Monk scripts.

e*Gate does not contain any pre-built Java validations, translations, enveloping, or acknowledgments. These business rules can be built in the Java Collaboration Editor GUI, which provides a user-friendly drag-and-drop front end for creating Java business rules.

e*Gate UN/EDIFACT Java ETDs have validations and translations, but a validation does not generate an acknowledgment transaction. Instead, it generates a string.

Note: *In e*Gate, translations are called Collaborations.*

Chapter 3

UN/EDIFACT Template Installation

This chapter tells you how to install UN/EDIFACT files and where to find them after installation.

Note: *If you are upgrading to version 4.5.2 from a previous version of this library, you must recompile all Collaborations that use these ETDs after installing the new version.*

3.1 System Requirements

The UN/EDIFACT ETD Library is available on the following operating systems:

- Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Windows 2000 (Japanese), Windows 2000 SP1 (Japanese), and Windows 2000 SP2 (Japanese)
- Windows NT 4 SP6a
- Windows NT 4 SP6a (Japanese)
- Solaris 2.6, 7, and 8
- Solaris 2.6, 7, and 8 (Japanese)
- AIX 4.3.3
- HP-UX 11.0, HP-UX 11i
- HP-UX 11.0 (Japanese)
- Compaq Tru64 V4.0F and V5.0A
- Red Hat Linux 6.2

To use the UN/EDIFACT ETD Library, you need the following:

- An e*Gate Participating Host, version 4.5 or later.
- A TCP/IP network connection.
- 1.79 GB of disk space to install all the UN/EDIFACT ETD Libraries on a Windows, UNIX, or Red Hat Linux system. The Java libraries are very large (for example: approximately 175 MB is required to install the Java and Monk libraries for d01a.

Note: *Additional disk space is required to process and queue the data that UN/EDIFACT processes; the amount necessary will vary based on the type and size of the data being processed.*

3.1.1 External System Requirements

The UN/EDIFACT ETD Library supports the following external systems:

- EDIFACT_d95a
- EDIFACT_d95b
- EDIFACT_d96a
- EDIFACT_d96b
- EDIFACT_d97a
- EDIFACT_d97b
- EDIFACT_d98a
- EDIFACT_d98b
- EDIFACT_d99a
- EDIFACT_d99b
- EDIFACT_d00a
- EDIFACT_d00b
- EDIFACT_d01a
- EDIFACT_d01b
- EDIFACT_v3
- EDIFACT_v4

3.2 Installing UN/EDIFACT ETD Templates

This section explains how to install the UN/EDIFACT ETD template files. Installing them performs the following:

- Installs the UN/EDIFACT add-on applications components.
- Installs add-on package files on the local client subdirectories and in the specified e*Gate Registry Repository.
- Installs the Java and Monk template files only in the specified Registry.

Table 8 lists which UN/EDIFACT ETD Libraries are located on which CD-ROM.

Table 8 Location of UN/EDIFACT ETD Libraries on CD-ROMs

CD-ROM 6		CD-ROM 7	
EDIFACT_d95a	EDIFACT_d98a	EDIFACT_d00a	EDIFACT_V3
EDIFACT_d95b	EDIFACT_d98b	EDIFACT_d00b	EDIFACT_V4
EDIFACT_d96a	EDIFACT_d99a	EDIFACT_d01a	
EDIFACT_d96b	EDIFACT_d99b	EDIFACT_d01b	
EDIFACT_d97a	EDIFACT_V3		
EDIFACT_d97b	EDIFACT_V4		

3.2.1 Windows NT and Windows 2000

Before installation

- Exit all Windows programs before running the setup program, including any anti-virus applications.
- You must have Administrator privileges to install this product.

Installation procedure

To install the UN/EDIFACT ETD templates on Windows

1. Log on to the workstation on which you want to install the UN/EDIFACT templates.
2. Insert the installation CD-ROM into the CD-ROM drive.

Note: *The UN/EDIFACT ETD Library add-ons can be found on CD-ROM 6 through CD-ROM 7.*

3. If the CD-ROM drive's Autorun feature is enabled, the setup application should launch automatically. Otherwise, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the file **setup.exe** on the CD-ROM drive.
4. The InstallShield setup application launches. Follow the on-screen instructions until you come to the **User Information** dialog box. Type your name and company name and then click **Next**.
5. When the **Please choose the product to install** dialog box appears, select the appropriate product and click **Next**.
6. The **Please choose the product to install** dialog box remains open with **Add-ons** selected. Click **Next**.
7. The **Check Setup Information** dialog box appears. Confirm your selections and click **Next**.

8 Follow the online prompts in the InstallShield Wizard to navigate through the introductory screens and to accept the license agreement.

9 When the **User Information** dialog box appears, type your name and company name, and then click **Next**.

10 The **Choose Destination Location** dialog box appears. The setup utility recommends the appropriate destination folder; we strongly recommend that you do not change the default. Click **Next**.

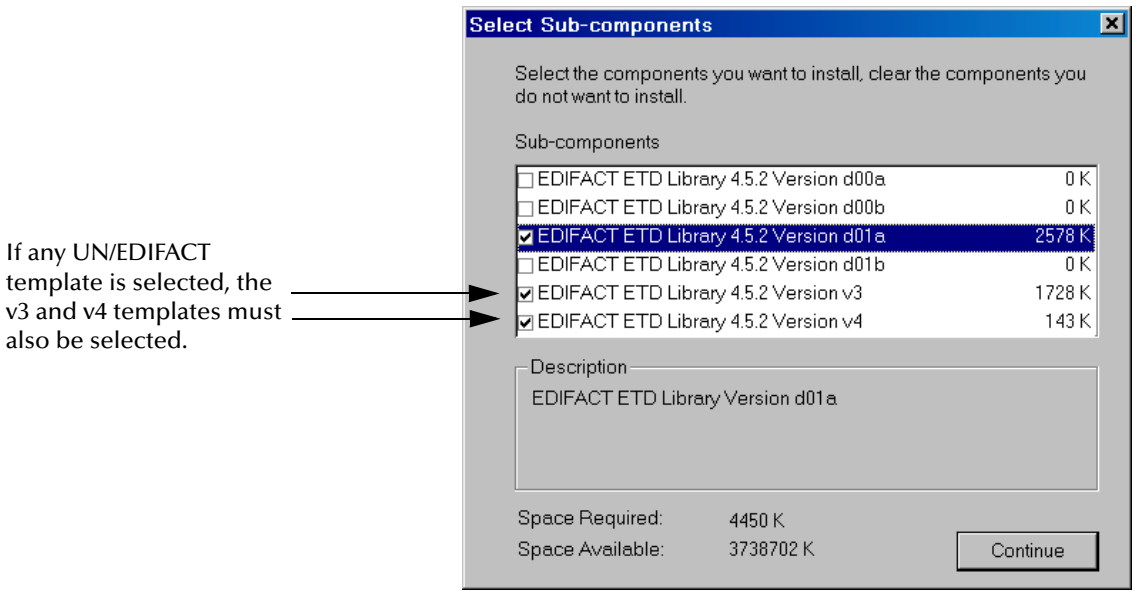
Note: *Spaces are not valid characters in e*Gate path names.*

11 When the **Select Components** dialog box appears, highlight **ETD Libraries**, and then click **Change**. This allows you to select the UN/EDIFACT ETD templates that you want to install.

Note: Do not select the check box.

12 When the **Select Sub-components** dialog box appears, select the UN/EDIFACT ETD templates that you want to install. Each library (for example: EDIFACT ETD Library 4.5.2 Version d01b) contains both the Java and the Monk templates. You must include the v3 and v4 templates.

Figure 2 Select Sub-components Dialog Box



Note: *Select the various check boxes and look in the **Description** region to see exactly what that particular library holds.*

You *must* install v3 and v4 because the information these envelopes hold is required for UN/EDIFACT messages. The UN/EDIFACT ETD Library Version v3 and v4 are two separate interchange envelope syntax versions (see **"Existing v3 Envelope**

Names" on page 42), such as the header envelope that accompanies every UN/EDIFACT message. The v3 templates only handle batch messages while the v4 templates can handle batch or interactive messages. As they do not change over time, only one template library has been created for each of them.

For more information on enveloping UN/EDIFACT files, see the United Nations Web site under "Additional information" on page 39.

13 After making your selections, click **Continue**, as shown above. The **Select Components** dialog box reappears. Click **Next** to continue.

14 Some add-ons have dependencies. If you select an add-on with dependencies, the **Check Add-ons Dependencies Information** dialog box opens. It informs you of additional components that will be selected if you have not already selected them. To change any settings, click **Back**. If satisfied, click **Next**.

15 The **Select Program Folder** dialog box appears. Choose a program folder (the default is **SeeBeyond eBusiness Integration Suite**) and click **Next**.

16 When the **Check Setup Information** dialog box appears, confirm your selections and click **Next**.

Note: *Be sure to install all of the template files in the suggested installation directory. The installation utility detects and suggests the appropriate installation directory.* ***Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.***

17 When prompted for the Registry Host on which these add-ons should be installed, enter the Registry Host's name (if installing to a Distributed Registry system, enter the primary Registry Host's name) and click **Next**.

18 The **Administrator Account Information** dialog box appears.

- In the **Username** box, type the name of the e*Gate "Administrator" user. Unless you have created additional "administrative" accounts, use the default name **Administrator**.
- In the **Password** and **Confirm** boxes, type and verify the appropriate password.

Note: *e*Gate user names and passwords are case-sensitive. For additional information on the Administrator and passwords, see the* ***e*Gate Integrator System Administration and Operations Guide****.*

When ready, click **Next**.

19 Select the platforms that the selected Registry Host(s) support and click **Next**.

The installation utility begins installing add-on files and committing them to the e*Gate Registry. The amount of time this process takes depends upon the number and size of add-ons you are installing.

20 Follow the on-screen prompts to complete the installation. Then repeat this process to install the UN/EDIFACT ETD Libraries that are located on the other CD-ROM. For details about e*Gate installation, see the *e*Gate Integrator Installation Guide*.

Note: *After completing the installation, the install wizard prompts you to restart your computer. You must restart your computer before using e*Gate. However, it is not necessary to restart your computer at this time.*

3.2.2 UNIX

When copying the generated files from Sun/UNIX to Windows, make sure the file names *remain lower-case*. Some Windows tools are particular about case conversions (for example, WinZip). File transfer protocol (FTP) does not have case problems.

Note: *The basic installation procedure for Compaq Tru64 systems is essentially the same as installing the UN/EDIFACT templates on a UNIX system.*

Before installation

You do not need root privileges to install the UN/EDIFACT ETD template libraries. Log in under the user name that you want to own the UN/EDIFACT ETD templates. Be sure that this user has sufficient privilege to create files in the e*Gate directory tree.

Installation procedure

To install the UN/EDIFACT ETD templates on a UNIX system

1 Log in on the workstation containing the CD-ROM drive, and insert the installation CD-ROM into the drive.

2 If necessary, mount the CD-ROM drive.

3 At the shell prompt, type:

cd /cdrom

4 Start the installation script by typing:

setup.sh

A menu of options appears.

5 Select the **e*Gate Addon Applications** option.

6 Follow the rest of the on-screen instructions to install the UN/EDIFACT template files. For details about e*Gate installation, see the *e*Gate Integrator Installation Guide*.

Note: *Be sure to install the template files in the suggested installation directory. The installation utility detects and suggests the appropriate installation directory.* ***Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.***

Additional information

For more information, see the United Nations URL on the World Wide Web, dealing with the UN/EDIFACT standards. This Internet site is:

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

Look under Section 5 in the left pane.

Note: *These URL directions reflect the United Nations Web site setup at this publication. If the site has changed setup and/or URL, see the current United Nations home page for directions.*

3.3 Files and Directories Created by the Installation

The UN/EDIFACT ETD templates installation on Windows NT/2000 places files within the e*Gate directory tree, on the Participating Host and committed to the **default** schema on the Registry Host.

The UNIX installation places the files in the same path locations and directories as the Windows NT/2000 installation. All file names are also the same. For more information about the UN/EDIFACT ETD Library, see **Chapter 4**.

UN/EDIFACT directories

UN/EDIFACT uses the term "directory" to refer to each version of the standard. This is not to be confused with the usual sense of file system 'directories' in the following discussion.

The e*Gate UN/EDIFACT ETD Library is organized into subdirectories as follows: There is a parent directory for all UN/EDIFACT versions (for example, in Monk: **eGate\Server\registry\repository\default\monk_scripts\templates\edifact**). Under that there is a directory for each version, such as edifact_d95a, edifact_d95b, ... edifact_d01b, and so on. Under each version directory there are subdirectories for segment definitions, v3, and v4 envelopes. All messages are in the version directory, and all segments are in the **templates** directory, as illustrated in **Table 9 on page 40**. In Java, the directory structure for the UN/EDIFACT versions is different than Monk (for example, in Java: **eGate\Server\registry\repository\default\etd\templates\edifact**). The Java UN/EDIFACT ETD directory structure is illustrated in **Table 10 on page 41**.

Table 9 UN/EDIFACT ETD Library Directory Structure for Monk

Directories	Directory Contents
edifact_d95a	
\templates	Segment definitions
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
edifact_d95b	
\templates	Segment definitions
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
edifact_d96a	
\templates	Segment definitions
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
----------	
edifact_d01a	
\templates	Segment definitions
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
edifact_d01b	
\templates	Segment definitions
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
edifact_v3	
CONTRL	Control definitions
\templates	v3 Envelope segment definitions
edifact_v4	
CONTRL	Control definitions
\templates	v4 Envelope segment definitions

When message, segment, and envelope data elements are stored as files they are preceded by or appended with the following characters:

- Messages are appended with **.ssc**.
- Segments are preceded with **es_** and appended with **.ssc**.
- v3 and v4 control envelopes are appended with **.ssc**.
- v3 and v4 segment envelopes are preceded with **es_** and appended with **.ssc**.

To learn more about these specialized template files, open and examine them, using the ETD Editor feature in e*Gate. For complete information on the purpose and function of each template, see the United Nations' World Wide Web site shown under **"Additional information" on page 39**.

Table 10 UN/EDIFACT ETD Library Directory Structure for Java

Directories	Directory Contents
edifact_d95a	
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
edifact_d95b	
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
edifact_d96a	
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
----------	
edifact_d01a	
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope
edifact_d01b	
\v3	Batch messages using v3 envelope
\v4	Batch and interactive messages using v4 envelope

Java messages are stored as files with the following characters:

- Messages are appended with **.xsc**.
- Compressed Java class files are appended with **.jar**.

Note: *When building Collaboration Rules scripts with Java ETDs, if there is data mapped to a field in a Java template and there are optional fields on the same level with no data mapped to them, the output will include delimiters for the optional fields.*

Chapter 4

UN/EDIFACT ETD Library

This chapter lists sample file and directory names in the UN/EDIFACT ETD Library; it also describes how to test data in specific files.

4.1 UN/EDIFACT Files and Directories

This section introduces the different types of UN/EDIFACT data elements, the files that hold the elements, and the directories that contain the files in SeeBeyond's UN/EDIFACT ETD Library. It also provides links to the tables in this chapter that list the data elements and files alphabetically and gives a breakdown of the files in each directory.

4.1.1 UN/EDIFACT Batch, Interactive, and Envelope File Names

UN/EDIFACT message names all have six alphabetic characters, while UN/EDIFACT segment names are all three characters long.

UN/EDIFACT data has:

- Six-letter message names
- Three-letter segment names

The messages and segments must be combined with the v3 and v4 envelopes in order for an electronic computer-to-computer transmission of data to take place. Each of these data elements, along with their function, is listed alphabetically in the tables in this section.

4.1.2 Existing v3 Envelope Names

A v3 envelope only contains batch envelope segments. The v3 envelope file names do not change very often. See the following tables for a listing of the v3 envelope names and their functions:

- **Figure 3 on page 43**
- **Table 11 on page 43**
- **Table 12 on page 43**

Note: These envelopes can be used with any version of the UN/EDIFACT ETD messages.

The v3 header and trailer envelope segments have set locations within the EDI structure, and must appear in the order as shown below. The lines on the left side of the diagram show how headers and footers work in pairs (see Figure 3).

Figure 3 v3 Envelope Segments

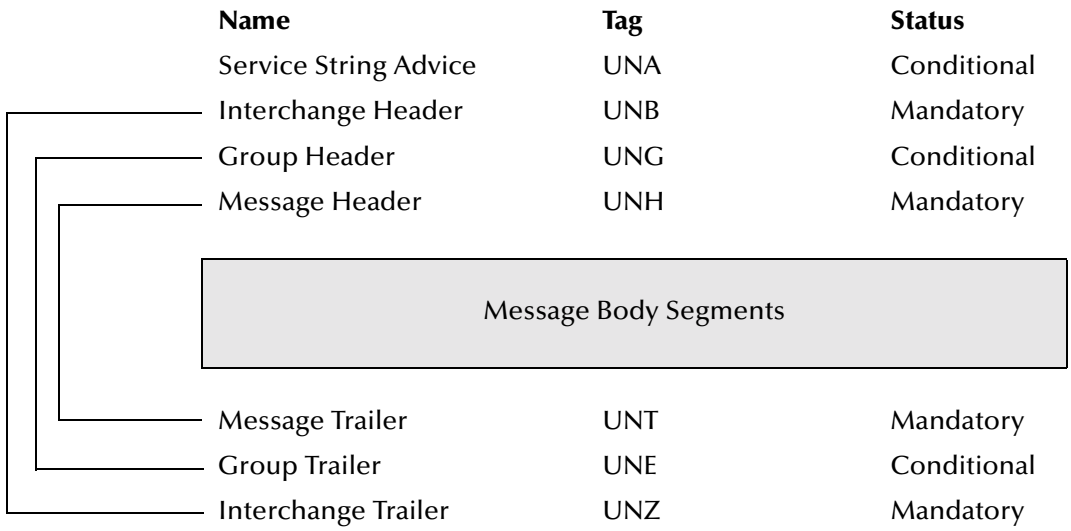


Table 11 v3 Control Message

v3 Control Message Name	v3 Control Message Function
CONTRL	Syntactically acknowledges or rejects, with error indication, a received interchange, functional group, or message.

Table 12 v3 Batch Segments

v3 Segment Name	v3 Segment Function
UCD	Data Element Error Indication
UCF	Functional Group Response
UCI	Interchange Response
UCM	Message Response
UCS	Segment Error Indication
UNA	Delimiter List
UNB	Interchange Header
UNE	Functional Group Trailer
UNG	Functional Group Header
UNH	Message Header
UNS	Section Control

Table 12 v3 Batch Segments (Continued)

v3 Segment Name	v3 Segment Function
UNT	Message Trailer
UNZ	Interchange Trailer

Note: *UNA is optional and seldom used.*

4.1.3 Existing v4 Envelope Names

A v4 envelope can have either batch or interactive envelope segments. *For an interactive envelope to be used, one or more dialogues must occur either concurrently or sequentially between two or more parties.* A dialogue consists of a pair of interleaved UN/EDIFACT interchanges:

- Initiator interchange
- Responder interchange

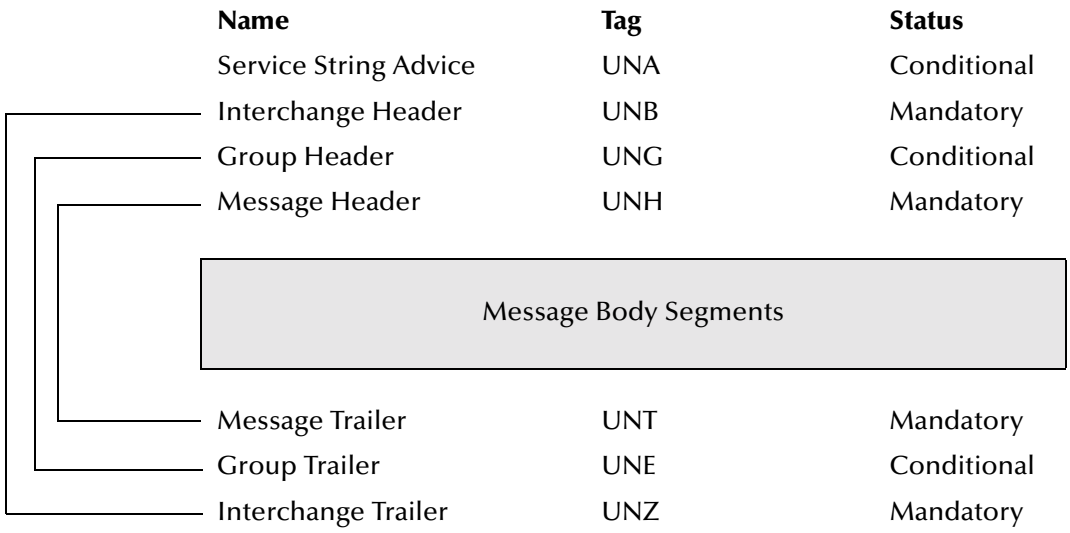
The v4 envelope file names do not change very often. See the following tables for a listing of the v4 batch and interactive envelope names and their functions:

- **Figure 4 on page 45**
- **Figure 5 on page 45**
- **Table 13 on page 46**
- **Table 14 on page 46**

Note: These envelopes can be used with any version of the UN/EDIFACT ETD messages.

The v4 batch header and trailer envelope segments have set locations within the EDI structure, and must appear in the order as shown below. The lines on the left side of the diagram show how headers and footers work in pairs (see Figure 4).

Figure 4 v4 Batch Envelope Segments



Note: *The v4 batch envelope segments are the same as the v3 batch envelope segments.*

The v4 interactive header and trailer envelope segments have set locations within the EDI structure, and must appear in the order as shown below. The lines on the left side of the diagram show how headers and footers work in pairs (see Figure 5).

Figure 5 v4 Interactive Envelope Segments

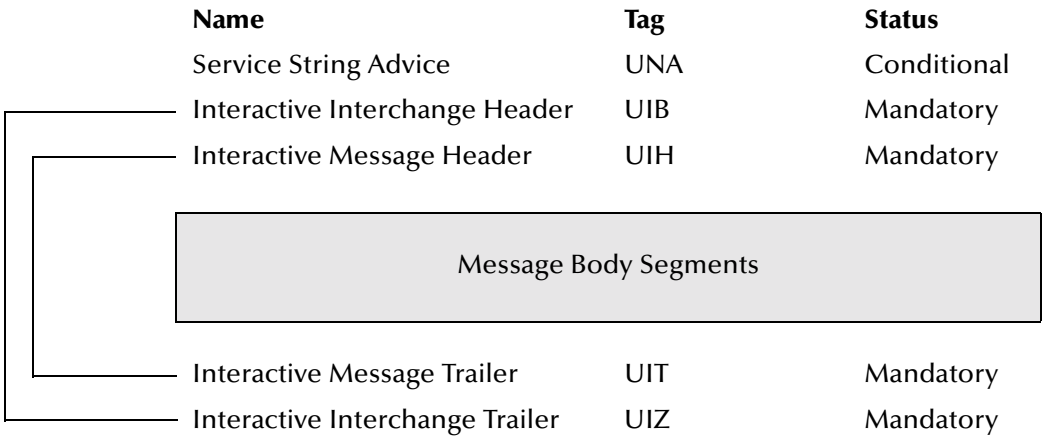


Table 13 v4 Control Message

v4 Control Message Name	v4 Control Message Function
AUTACK	**Authentication function**. Authenticates sent, or provides secure acknowledgment of received interchanges, groups, messages, or packages. It is sent by either the originator of at least one UN/EDIFACT structure or by a party authorized by the originator to act in its behalf. **Acknowledgment function**. An acknowledgment message sent by either the recipient of one or more secure UN/EDIFACT structures or by a party authorized by the recipient to act in its behalf. Note that the acknowledgment function applies only to UN/EDIFACT structures that have been secured.
CONTRL (for batch EDI)	Syntactically acknowledges or rejects, with error indication, a received interchange, functional group, message, or package. A maximum of two CONTRL messages, the first of which is optional, can be sent in response to a received interchange: ♦ After an interchange, the first message provides an indication of the receipt. ♦ After the syntax check of the subject interchange, the second message reports the action taken.
KEYMAN	Provides certificate management and a security key. There are two types of keys: ♦ A secret key that is used with symmetric algorithms. ♦ A public or private key used with asymmetric algorithms.

Table 14 v4 Segments

v4 Segment Name	v4 Segment Function
UCD	Data Element Error Indication
UCF	Group Response
UCI	Interchange Response
UCM	Message / Package Response
UCS	Segment Error Indication
UGH	Anti-Collision Segment Group Header
UGT	Anti-Collision Segment Group Trailer
UIB	Interactive Interchange Header
UIH	Interactive Message Header
UIR	Interactive Status
UIT	Interactive Message Trailer

Table 14 v4 Segments (Continued)

v4 Segment Name	v4 Segment Function
UIZ	Interactive Interchange Trailer
UNA	Delimiter List
UNB	Interchange Header
UNE	Group Trailer
UNG	Group Header
UNH	Message Header
UNO	Object Header
UNP	Object Trailer
UNS	Section Control
UNT	Message Trailer
UNZ	Interchange Trailer
USA	Security Algorithm
USB	Secured Data Identification
USC	Certificate
USD	Data Encryption Header
USE	Security Message Relation
USF	Key Management Function
USH	Security Header
USL	Security List Status
USR	Security Result
UST	Security Trailer
USU	Data Encryption Trailer
USX	Security References
USY	Security On References

Note: *UNA is optional and seldom used.*

4.2 Using the Java Tester to Test an ETD

e*Gate supports a Java Tester that tests ETDs to ensure that data maps to an ETD correctly. The Tester maps the input data to an ETD, and generates an output file containing data from the mapped ETD. When using the Java Tester, the output should mirror the input.

The test performs the following steps:

1. Maps (unmarshals) the input data into the ETD structure.
2. Validates the data (return type = string).
3. Generates (marshals) the output data from the ETD mapped with input data in Step 1.

Five **.jar** files are used during the test:

- **c:\eGate\client\etd\templates\edifact\edifact_d01a\v3\EDF_D01A_BAPLTE_BaypPlanTotalNumbMess.jar**
  (the actual ETD structure)
- **c:\eGate\client\etd\templates\edifact\edifact_d01a\v3\EDF_D01A_AllSegsComs.jar**
  (combines all the segments and composites for this version and release)

Note: *Users need to use* ***stcregutil.exe*** *to retrieve both the* ***<EDF_D01A_BAPLTE_BaypPlanTotalNumbMess.jar>*** *and the shared segments/composites* ***.jar*** *file from the Registry Host to the Participating Host where the command-line tester or the Collaboration Rules Editor GUI runs. For example:*

stcregutil.exe -rs default -rh <registry hostname> -un <user name> - up <password> -fr etd\templates\edifact\edifact_d00a\v3\[EDF_D01A_AllSegsComs.jar]/[EDF__D01A_BAPLTE_BaypPlanTotalNumbMess.jar]

- **c:\eGate\client\classes\stcjcs.jar**
- **c:\egate\client\bin\java\xerces.jar**
- **c:\egate\client\bin\java\gnu-regexpj-1.1.1.jar**
  (the top level UN/EDIFACT Java class)

During the test, the **classpath** must be set to include all five **.jar** files.

To change the classpath

- Type: **set classpath=**

Note: *The changes to classpath can be made before running the utility or at the same time the utility runs using the "-classpath" option of the "java" command.*

4.2.1 Running the Test

The test only accepts data with envelopes. When running the test, you may set the classpath and run the command (with three parameters) at the same time. Also, you may change the classpath environment variable before running the test (see above). Use a semicolon (;) in Windows to separate the strings; use a colon (:) in UNIX to separate the strings.

For example:

java -classpath c:\eGate\Server\registry\repository\default\etd\templates\edifact\edifact_d01a\v3\EDF_D01A_BAPLTE_BaypPlanTotalNumbMess.jar;c:\eGate\Server\registry\repository\default\etd\templates\edifact\edifact_d01a\v3\EDF_D01A_AllSegsComs.jar;c:\eGate\client\classes\stcjcs.jar;c:\egate\client\bin\java\xerces.jar;c:\egate\client\bin\java\gnu-regexp-1.1.1.jar com.stc.edifact_v3_d95B.EDF_D01A_BAPLTE_BaypPlanTotalNumbMessOuter <*input data file name*> <*output data file name*>

Where

- **EDF_D01A_BAPLTE_BaypPlanTotalNumbMess.jar** = name of input **.jar** file
- **EDF_D01A_BAPLTE_BaypPlanTotalNumbMessOuter** = name of node
- <***input data file name***> = the name of an existing input data file
- <***output data file name***> = the name of the file to which the output is written (it does not have to be an existing file)

The results of the test are printed to the output data file. This output data content should be the same as the input data, but the format may be different for certain fields. For example:

- If there is a date, the output file might have an eight-digit date (“20010420”) rather than a six-digit date (“010420”).
- The number of trailing zeros after a decimal point for a double field might differ (for example: input of “10.00” might output as “10”).

4.2.2 Using the ETD Editor to View an .xsc File

Use the ETD Editor (make sure that Java has been selected as the default editor) to load and view an .xsc file. You cannot edit the file; however, you can view the ETD’s structure. In an UN/EDIFACT**. xsc** file, the elements always begin with an “E.”

For example:

E0001_1_SyntIden (the first element of the composite “S001”)

Note: *If an item starts with a Loop in an* ***.xsc*** *file, it is a segment loop (for example:* ***LoopRFF_5_Refe****).*

To open the .xsc file

Remember, Java must be selected as the default editor.

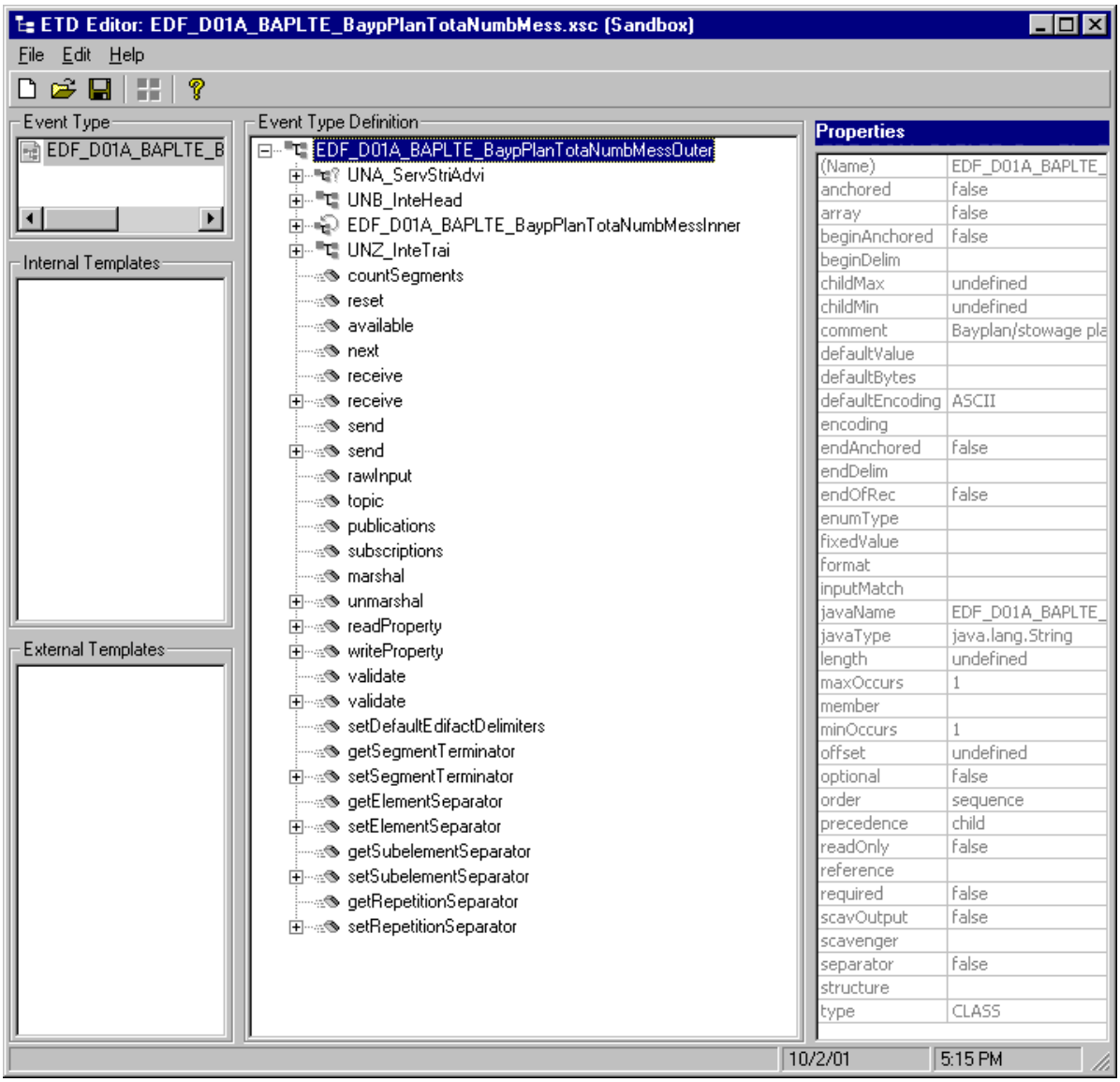
1 In the ETD Editor, choose **File** and then click **Open**.

2 Navigate to the appropriate directory.
For example: **etd\templates\edifact\edifact_d01a\v3**

3 Select an **.xsc** file and click **Open**. See Figure 6.

Note: *The root name of the node carries "Outer" on the end of it, the same value as was used in the command-line utility.*

Figure 6 Example of an .xsc File in the ETD Editor



4.2.3 Delimiters in an .xsc File

UN/EDIFACT uses six delimiters. These delimiters, which the user can get and set from Java, are:

- Component data element separator
- Data element separator
- Decimal notation
- Release indicator
- Reserved for future use
- Segment terminator

To see an example of these elements, see the **.xsc** example shown in Figure 6. For example: **setDefaultEdifactDelimiters**.

4.2.4 Using the Collaboration Rules Editor to Validate an .xsc file

The elements that are part of an **.xsc** file can be dragged and dropped when two or more **.xsc** files are opened in the Collaboration Rules Editor (see the *e*Gate Integrator User's Guide* for more information). A field in the Source pane can be dragged to a field in the Destination Events pane. This action, when highlighted in the Business Rules pane, displays the rule in the Rule Properties pane.

The "validate" method nodes in an **.xsc** file can be used to validate an UN/EDIFACT message at run time. The methods return a string containing description(s) about any invalid data elements, segments, segment loops, envelopes, et cetera. Although validation should be used to ensure that data is good, be aware that validation significantly impacts performance.

Before compiling a Collaboration

The classpath for a Collaboration must be set before the Collaboration is compiled. This ensures that the v3 batch and v4 batch file (**EDF_<*D01A*>_AllSegsComs.jar**), which is necessary if the UN/EDIFACT file is batch, or the v4 interactive file (**EDF_<*D01A*>_V4Interactive_AllSegsComs.jar**), which is necessary if the UN/EDIFACT file is interactive, remains with the main **.jar** file (for example: **EDF_D01A_BAPLTE_BaypPlanTotalNumbMess.jar**).

To set the classpath

1. With a Collaboration Rule open in the Collaboration Rules Editor, select **Tools** and then click **Options** on the menu bar.

Note: *See the **e*Gate Integrator User's Guide** for information on how to create and open Collaboration Rules.*

2. The **Options** dialog box opens with the **Classpath** tab selected. Click **Add File**.

3 Navigate to the appropriate client directory (such as **eGate\client\etd\templates\edifact_d01a\V3**), select the **.jar** file (for example: **EDF_<*D01A*>_AllSegsComs.jar**) and click **Open**.

Note: *You must have previously copied the appropriate **.jar** file (for example: **EDF_<*D01A*>_AllSegsComs.jar**) to this client directory to make it available for selection.*

4 Click **OK** to return to the Collaboration Rules Editor.

4.3 UN/EDIFACT Java Methods

The templates in the UN/EDIFACT ETD Library contain the methods that allow you to set and get the delimiters, which in turn extend the functionality of the EDIFACT ETD Library.

4.3.1 Methods to Set or Get Delimiters

Each **.xsc** file (for example: **com.stc.edifact_v3_d95B.EDF_...Outer,** which could represent **com.stc.edifact_v3_d95B.EDF_D01A_BAPLTE_BaypPlanTotalNumbMessOuter**) serves as the class for the following methods. Use these methods to set or get the default delimiters for each UN/EDIFACT ETD Template Library.

The **com.stc.edifact_v3_d95B.EDF_...Outer** class extends **com.stc.jcsre.EDFETDImpl** and implements **com.stc.jcsre.ETD**.

The **com.stc.edifact_v3_d95B.EDF_...Outer** methods are:

- **setDefaultEdifactDelimiters** on page 53
- **getSegmentTerminator** on page 53
- **setSegmentTerminator** on page 54
- **getElementSeparator** on page 55
- **setElementSeparator** on page 55
- **getSubelementSeparator** on page 56
- **setSubelementSeparator** on page 56
- **getRepetitionSeparator** on page 57
- **setRepetitionSeparator** on page 58

The UN/EDIFACT ETD Library also includes the following custom Java methods for testing the validation Collaboration:

- **validate (no parameters)** on page 58
- **validate (one parameter)** on page 59

setDefaultEdifactDelimiters

Description

Sets the default UN/EDIFACT delimiters.

Syntax

```
public void setDefaultEdifactDelimiters()
```

Parameters

Name	Type	Description
setDefaultEdifactDelimiters	METHOD	Sets the current delimiters to the default UN/EDIFACT delimiters: - segmentTerminator = ' - elementSeparator = + - subelementSeparator = : - repetitionSeparator = *

setDefaultEdifactDelimiters Constants

None

Returns

Void

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
myETD.setDefaultEdifactDelimiters();
```

getSegmentTerminator

Description

Gets the segmentTerminator character.

Syntax

```
public char getSegmentTerminator()
```

Parameters

Name	Type	Description
getSegmentTerminator char.	METHOD	Gets the segmentTerminator character.

getSegmentTerminator Constants

None

Returns

char
Returns the segment terminator character.

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
char segTerm=myETD.getSegmentTerminator();
```

setSegmentTerminator

Description

Sets the segmentTerminator character.

Syntax

```
public void setSegmentTerminator(char c)
```

Parameters

Name	Type	Description
setSegmentTerminator char.	METHOD	Sets the segmentTerminator character.

setSegmentTerminator Constants

None

Returns

Void

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
char c='~';
myETD.setSegmentTerminator(c);
```

getElementSeparator

Description

Gets the elementSeparator character.

Syntax

```
public char getElementSeparator()
```

Parameters

Name	Type	Description
getElementSeparator char.	METHOD	Gets the elementSeparator character.

getElementSeparator Constants

None

Returns

char
Returns the element separator character.

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
char elmSep=myETD.getElementSeparator();
```

setElementSeparator

Description

Sets the elementSeparator character.

Syntax

```
public void setElementSeparator(char c);
```

Parameters

Name	Type	Description
setElementSeparator char.	METHOD	Sets the elementSeparator character.

setElementSeparator Constants

None

Returns

Void

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
char c='+';
myETD.setElementSeparator(c);
```

getSubelementSeparator

Description

Gets the subelementSeparator character.

Syntax

```
public char getSubelementSeparator()
```

Parameters

Name	Type	Description
getSubelementSeparator char.	METHOD	Gets the SubelementSeparator character.

getSubelementSeparator Constants

None

Returns

char
Returns the getSubelement character.

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
char subeleSep=myETD.getSubelementSeparator();
```

setSubelementSeparator

Description

Sets the SubelementSeparator character.

Syntax

```
public void setSubelementSeparator(char c)
```

Parameters

Name	Type	Description
setSubelementSeparator char.	METHOD	Sets the SubelementSeparator character.

setSubelementSeparator Constants

None

Returns

Void

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
char c=':';
myETD.setSubelementSeparator(c);
```

getRepetitionSeparator

Description

Gets the RepetitionSeparator character.

Syntax

```
public char getRepetitionSeparator()
```

Parameters

Name	Type	Description
getRepetitionSeparator char.	METHOD	Gets the RepetitionSeparator character.

getRepetitionSeparator Constants

None

Returns

char

Returns the getRepetitionSeparator character.

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
char repSep=myETD.getRepetitionSeparator();
```

setRepetitionSeparator

Description

Sets the RepetitionSeparator character.

Syntax

```
public void setRepetitionSeparator(char c)
```

Parameters

Name	Type	Description
setRepetitionSeparator char.	METHOD	Sets the RepetitionSeparator character.

setRepetitionSeparator Constants

None

Returns

Void

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
char c='*';
myETD.setRepetitionSeparator(c);
```

validate (no parameters)

Description

Validates the ETD content in memory.

Syntax

```
public String validate()
```

Parameters

None.

validate Constants

None.

Returns

String

Throws

None.

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
string msg=myETD.validate();
```

validate (one parameter)

Description

Validates the ETD content, either immediately after unmarshaling or in memory. When the parameter is false, this method works in the same way as validate (with no parameters). When the parameter is true, this can be used to validate length information in the input data file.

Syntax

```
public String validate(boolean original)
```

Parameters

Name	Type	Description
original	boolean	If true, validates the ETD content right after unmarshaling. If false, validates the ETD in memory.

validate Constants

None.

Returns

String

Throws

None

Examples

```
com.stc.edifact_v3_d95B.EDF_..._...Outer myETD=new com.stc.edifact_v3_d95B.
EDF_..._Outer();
......
......
string msg=myETD.validate(true);
```

Index

Symbols

A

B

C

D

E

I

J

K

L

M

P

R

S

U

V

W

X