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.

© 2004 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 This work is protected as an unpublished work under the copyright laws.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 20040603180353.
Contents

List of Figures 6

List of Tables 7

Chapter 1

Introduction 8

Overview 8

Intended Reader 8

Compatible Systems 8

Document Organization 9

Writing Conventions 9

Additional Conventions 9

Supporting Documents 10

SeeBeyond Web Site 10

UN/ECE Web Site 10

Chapter 2

Overview of UN/EDIFACT 11

UN/EDIFACT Components 11

Message Structure 12

Messages 13

Segment Table 18

Loops 19

Envelopes 20

UNA segment 20

Control messages 20

Delimiters 21

OTD Libraries 21

UN/EDIFACT Versus X12 22

Security 22

Examples of EDI Usage 22

Overview of EDI Payments Processing 22
Contents

Exchange of remittance information 23
Routing of remittance information 23
Exchange of payment orders 23
Functions a payment must perform 24
Formats for transporting a payment 24
Issuance of a payment order 24
Payment-Related EDI Transactions 25
  X12 25
  UN/EDIFACT 25
Understanding Enveloping Scenarios 26
  Point-to-point scenario 28
  End-to-end scenario 28
Payment Acknowledgments 29

Key Parts of EDI Processing Logic 30
  Structures 30
  Validations, Translations, Enveloping, Acknowledgments 30

Chapter 3

Installation 31
  System Requirements 31
  Installation Procedure 32
    Uploading to the Repository 32
    Refreshing Enterprise Designer 33
  UN/EDIFACT Library Templates 34
    UN/EDIFACT OTDs 34
    Transaction Template Names 34
      eGate Project Explorer Display of UN/EDIFACT OTDs 35

Chapter 4

UN/EDIFACT OTD Library 36
  UN/EDIFACT Files and Directories 36
    UN/EDIFACT Batch, Interactive, and Envelope File Names 36
    Existing v3 Envelope Names 36
    Existing v4 Envelope Names 38

Chapter 5

Working With the EDIFACT OTDs 42
  Importing .jar Files 42
  Viewing an EDIFACT OTD in the OTD Editor 43
  Setting the Delimiters 46
  Methods for Getting and Setting 46
Bean Nodes for Getting and Setting Data 47
Bean Nodes for Getting Errors and Results 47
Using Validation in the Java Collaboration Editor 48
Creating a Collaboration Rule to Validate an OTD 48
Alternative Formats: ANSI and XML 49
XML Format for EDIFACT 49
Possible Differences in Output When Using Pass-Through 52
Limitations of EDIFACT OTDs 52
Memory Requirements 52
Delayed Unmarshaling 53
Errors and Exceptions 53
Special Methods for Error Classes 54

Chapter 6

Java Methods for EDIFACT OTDs 55

Using the OTD Editor to View and Test an OTD 55
To open an OTD 55
To test OTDs against sample data using the OTD Editor 56
Delimiters 56
Using the Collaboration Rules Editor to Validate an OTD 57

Methods 57
Java Methods to Set or Get Delimiters 58
setDefaultEdifactDelimiters 58
getSegmentTerminator 58
setSegmentTerminator 59
getElementSeparator 60
setElementSeparator 60
getSubelementSeparator 61
setSubelementSeparator 61
getRepetitionSeparator 62
setRepetitionSeparator 63
validate 63

Index 65
List of Figures

Figure 1  Example Payment Scenario 27
Figure 2  Update Center Wizard: Select Modules to Install 33
Figure 3  Some of the Transaction Set Structures for EDIFACT Version D00A 34
Figure 4  v3 Envelope Segments 37
Figure 5  v4 Batch Envelope Segments 39
Figure 6  v4 Interactive Envelope Segments 39
Figure 7  Importing sefimpl.jar 43
Figure 8  X12 270 Transaction in the OTD Editor 45
Figure 9  Accessing a Method in an X12 OTD 48
Figure 10  Accessing the performValidation Method from the Root Node 49
Figure 11  XML EDIFACT DTD 50
Figure 12  EDIFACT 997 Functional Acknowledgment—XML 51
Figure 13  EDIFACT 997 Functional Acknowledgment—ANSI Format 51
Figure 14  Setting the Maximum Heap Size 52
Figure 15  Example of an .xsc File in the OTD Editor 56
List of Tables

Table 1  Writing Conventions 9
Table 2  Batch Messages Defined in Version D00A 13
Table 3  Comparison Between X12 and UN/EDIFACT Envelopes 20
Table 4  Comparison of X12 to UN/EDIFACT: Payment Order/Remittance Advice 26
Table 5  Other Related Transactions 26
Table 6  Sample X12 and UN/EDIFACT Headers 27
Table 7  Types of UN/EDIFACT and X12 Acknowledgments 29
Table 8  Key Terms of EDI Processing 30
Table 9  UN/EDIFACT Versions Supported 31
Table 10  OTD Library Hierarchy in Project Explorer 35
Table 11  v3 Control Message 37
Table 12  v3 Batch Segments 37
Table 13  v4 Control Message 40
Table 14  v4 Segments 40
Chapter 1

Introduction

This chapter introduces you to the UN/EDIFACT OTD Library User’s Guide.

1.1 Overview

Each of the eGate Object Type Definition (OTD) libraries contains sets of pre-built structures for industry-standard formats. The UN/EDIFACT OTD Library is one of the products within the SeeBeyond Technology Corporation™ (SeeBeyond™) ICAN Suite. The OTD library contains message definitions for EDIFACT messages. This document gives a brief overview of both EDIFACT and the EDIFACT message structures that are provided, and provides information on installing and using the OTD library.

1.2 Intended Reader

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the SeeBeyond ICAN Suite (such as eGate Integrator and eXchange Integrator), to have familiarity with Windows operations and administration, and to be thoroughly familiar with Microsoft Windows graphical user interfaces.

1.3 Compatible Systems

The UN/EDIFACT OTD Library is available on the following platforms:

- Sun Solaris 8 and Solaris 9
- IBM AIX 5L Version 5.1 and AIX 5L Version 5.2
- HP-UX 11.0 and HP-UX 11i (PA-RISC)
- HP Tru64 UNIX Version 5.1A
- Red Hat Linux 8 (Intel Version) and Linux Advanced Server 2.1 (Intel version)
1.4 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 “Installation”** explains how to install UN/EDIFACT files and where to find them after installation.

- **Chapter 4 “UN/EDIFACT OTD Library”** lists sample file and directory names in the UN/EDIFACT OTD Library.

- **Chapter 5 “Working With the EDIFACT OTDs”** provides instructions and examples on how to load, view, and test EDIFACT OTDs.

- **Chapter 6 “Java Methods for EDIFACT OTDs”** lists and explains the bean nodes and Java methods that can be used to extend the functionality of the OTDs in the library.

1.5 Writing Conventions

The following writing conventions are observed throughout this document.

**Table 1** Writing Conventions

<table>
<thead>
<tr>
<th>Text</th>
<th>Convention</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names of buttons, files, menus and menu items, icons, parameters,</td>
<td><strong>Bold</strong> text</td>
<td>▪ Select the <em>logicalhost.exe</em> file.</td>
</tr>
<tr>
<td>variables, methods, and objects</td>
<td></td>
<td>▪ On the <strong>File</strong> menu, click <strong>Exit</strong>.</td>
</tr>
<tr>
<td>Command-line arguments, code samples</td>
<td>Fixed font.</td>
<td>▪ Enter the <em>timeout</em> value.</td>
</tr>
<tr>
<td>Hypertext links</td>
<td><strong>Bold italic</strong></td>
<td>▪ Use the <em>getClassName()</em> method.</td>
</tr>
<tr>
<td>bootstrap -f -p <strong>password</strong></td>
<td></td>
<td>▪ For more information, see “Writing Conventions” on page 9.</td>
</tr>
</tbody>
</table>

**Additional Conventions**

**Windows Systems**

For the purposes of this guide, references to “Windows” will apply to Microsoft Windows Server 2003, Windows XP, and Windows 2000.
Path Name Separator

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

1.6 Supporting Documents

The following SeeBeyond documents provide additional information about eGate and the ICAN system:

- SeeBeyond ICAN Suite Installation Guide
- SeeBeyond ICAN Suite Primer
- SeeBeyond ICAN Suite Deployment Guide
- eGate Integrator User’s Guide
- eGate Integrator Tutorial
- eGate Integrator System Administration Guide
- eXchange Integrator User’s Guide
- HIPAA OTD Library User’s Guide
- X12 OTD Library User’s Guide

You can also refer to the appropriate Microsoft Windows or UNIX documents, if necessary.

1.7 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

1.8 UN/ECE Web Site

UN/ECE, the United Nations Economic Commission of Europe, is one of the five regional commissions of the United Nations. The UN/ECE Web site contains technical information concerning rules, standards, recent UN/EDIFACT directories, syntax, and so forth. The site’s URL is:

http://www.unece.org/trade/untdid/welcome.htm
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 OTD Library allows eGate and eXchange customers to easily visualize the structures within a graphical user interface and to build up business rules (Collaborations) through drag and drop technology.

OTDs

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

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:


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.

**UN/EDIFACT 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 eGate, a message structure is called an Object Type Definition (OTD). Each message structure (OTD) consists of the following:

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

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

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

The 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 EDIFACT OTD Library 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.

eGate Integrator uses Object Type Definitions based on EDIFACT message structures to verify that the data in the messages coming in or going out is in the correct format. There is a message structure for each UN/EDIFACT transaction.

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

As an example, Table 2 below lists the batch messages, along with their function, that are defined in version D00A:

Table 2 Batch Messages Defined in Version D00A

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>APERAK</td>
<td>Application error and acknowledgement message</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>Authorization message</td>
</tr>
<tr>
<td>BALANC</td>
<td>Balance message</td>
</tr>
<tr>
<td>BANSTA</td>
<td>Banking status message</td>
</tr>
<tr>
<td>BAPLIE</td>
<td>Bayplan/stowage plan occupied and empty locations message</td>
</tr>
<tr>
<td>BAPLTE</td>
<td>Bayplan/stowage plan total numbers message</td>
</tr>
<tr>
<td>BERMAN</td>
<td>Berth management message</td>
</tr>
<tr>
<td>BMISRM</td>
<td>Bulk marine inspection summary report message</td>
</tr>
<tr>
<td>BOPBNK</td>
<td>Bank transactions and portfolio transactions report message</td>
</tr>
<tr>
<td>BOPCUS</td>
<td>Balance of payment customer transaction report message</td>
</tr>
<tr>
<td>BOPDIR</td>
<td>Direct balance of payment declaration message</td>
</tr>
<tr>
<td>BOPINF</td>
<td>Balance of payment information from customer message</td>
</tr>
<tr>
<td>BUSCRD</td>
<td>Business credit report message</td>
</tr>
<tr>
<td>CALINF</td>
<td>Vessel call information message</td>
</tr>
<tr>
<td>CASINT</td>
<td>Request for legal administration action in civil proceedings message</td>
</tr>
<tr>
<td>CASRES</td>
<td>Legal administration response in civil proceedings message</td>
</tr>
<tr>
<td>CHACCO</td>
<td>Chart of accounts message</td>
</tr>
<tr>
<td>CLASET</td>
<td>Classification information set message</td>
</tr>
<tr>
<td>CNTCND</td>
<td>Contractual conditions message</td>
</tr>
<tr>
<td>COACSU</td>
<td>Commercial account summary message</td>
</tr>
<tr>
<td>COARRI</td>
<td>Container discharge/loading report message</td>
</tr>
<tr>
<td>CODECO</td>
<td>Container gate-in/gate-out report message</td>
</tr>
<tr>
<td>CODENO</td>
<td>Permit expiration/clearance ready notice message 5</td>
</tr>
<tr>
<td>COEDOR</td>
<td>Container stock report message</td>
</tr>
<tr>
<td>COHAOR</td>
<td>Container special handling order message</td>
</tr>
<tr>
<td>COLREQ</td>
<td>Request for a documentary collection message</td>
</tr>
<tr>
<td>COMDIS</td>
<td>Commercial dispute message</td>
</tr>
<tr>
<td>CONAPW</td>
<td>Advice on pending works message</td>
</tr>
<tr>
<td>CONDPV</td>
<td>Direct payment valuation message</td>
</tr>
<tr>
<td>CONDRA</td>
<td>Drawing administration message</td>
</tr>
</tbody>
</table>
Table 2  Batch Messages Defined in Version D00A  (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDRO</td>
<td>Drawing organization message</td>
</tr>
<tr>
<td>CONEST</td>
<td>Establishment of contract message</td>
</tr>
<tr>
<td>CONITT</td>
<td>Invitation to tender message</td>
</tr>
<tr>
<td>CONPVA</td>
<td>Payment valuation message</td>
</tr>
<tr>
<td>CONQVA</td>
<td>Quantity valuation message</td>
</tr>
<tr>
<td>CONRPW</td>
<td>Response of pending works message</td>
</tr>
<tr>
<td>CONTEN</td>
<td>Tender message</td>
</tr>
<tr>
<td>CONWQD</td>
<td>Work item quantity determination message</td>
</tr>
<tr>
<td>COPAYM</td>
<td>Contributions for payment</td>
</tr>
<tr>
<td>COPARN</td>
<td>Container announcement message</td>
</tr>
<tr>
<td>COPINO</td>
<td>Container pre-notification message</td>
</tr>
<tr>
<td>COPRAR</td>
<td>Container discharge/loading order message</td>
</tr>
<tr>
<td>COREOR</td>
<td>Container release order message</td>
</tr>
<tr>
<td>COSTCO</td>
<td>Container stuffing/stripping confirmation message</td>
</tr>
<tr>
<td>COSTOR</td>
<td>Container stuffing/stripping order message</td>
</tr>
<tr>
<td>CREADV</td>
<td>Credit advice message</td>
</tr>
<tr>
<td>CREEXT</td>
<td>Extended credit advice message</td>
</tr>
<tr>
<td>CREMUL</td>
<td>Multiple credit advice message</td>
</tr>
<tr>
<td>CUSCAR</td>
<td>Customs cargo report message</td>
</tr>
<tr>
<td>CUSDEC</td>
<td>Customs declaration message</td>
</tr>
<tr>
<td>CUSEXP</td>
<td>Customs express consignment declaration message</td>
</tr>
<tr>
<td>CUSPDE</td>
<td>Periodic customs declaration message</td>
</tr>
<tr>
<td>CUSREP</td>
<td>Customs conveyance report message</td>
</tr>
<tr>
<td>CUSRRES</td>
<td>Customs response message</td>
</tr>
<tr>
<td>DEBADV</td>
<td>Debit advice message</td>
</tr>
<tr>
<td>DEBMUL</td>
<td>Multiple debit advice message</td>
</tr>
<tr>
<td>DEBREC</td>
<td>Debts recovery message</td>
</tr>
<tr>
<td>DELFOR</td>
<td>Delivery schedule message</td>
</tr>
<tr>
<td>DELIJIT</td>
<td>Delivery just in time message</td>
</tr>
<tr>
<td>DESADV</td>
<td>Despatch advice message</td>
</tr>
<tr>
<td>DESTIM</td>
<td>Equipment damage and repair estimate message</td>
</tr>
<tr>
<td>DGRECA</td>
<td>Dangerous goods recapitulation message</td>
</tr>
<tr>
<td>DIRDEB</td>
<td>Direct debit message</td>
</tr>
<tr>
<td>DIRDEF</td>
<td>Directory definition message</td>
</tr>
<tr>
<td>DMRDEF</td>
<td>Data maintenance status report/query message</td>
</tr>
</tbody>
</table>
### Table 2: Batch Messages Defined in Version D00A (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMSTAT</td>
<td>Data maintenance status report/query message</td>
</tr>
<tr>
<td>DOCADV</td>
<td>Documentary credit advice message</td>
</tr>
<tr>
<td>DOCAMA</td>
<td>Advice of an amendment of a documentary credit message</td>
</tr>
<tr>
<td>DOCAMI</td>
<td>Documentary credit amendment information message</td>
</tr>
<tr>
<td>DOCAMR</td>
<td>Request for an amendment of a documentary credit message</td>
</tr>
<tr>
<td>DOCAPP</td>
<td>Documentary credit application message</td>
</tr>
<tr>
<td>DOCARE</td>
<td>Response to an amendment of a documentary credit message</td>
</tr>
<tr>
<td>DOCINF</td>
<td>Documentary credit issuance information message</td>
</tr>
<tr>
<td>ENTREC</td>
<td>Accounting entries message</td>
</tr>
<tr>
<td>FINCAN</td>
<td>Financial cancellation message</td>
</tr>
<tr>
<td>FINPAY</td>
<td>Multiple interbank funds transfer message</td>
</tr>
<tr>
<td>FINSTA</td>
<td>Financial statement of an account message</td>
</tr>
<tr>
<td>GENRAL</td>
<td>General purpose message</td>
</tr>
<tr>
<td>GESMES</td>
<td>Generic statistical message</td>
</tr>
<tr>
<td>HANMOV</td>
<td>Cargo/goods handling and movement message</td>
</tr>
<tr>
<td>IFCSUM</td>
<td>Forwarding and consolidation summary message</td>
</tr>
<tr>
<td>IFTCCA</td>
<td>Forwarding and transport shipment charge calculation message</td>
</tr>
<tr>
<td>IFTDGN</td>
<td>Dangerous goods notification message</td>
</tr>
<tr>
<td>IFTFCC</td>
<td>International transport freight costs and other charges message</td>
</tr>
<tr>
<td>IFTIAG</td>
<td>Dangerous cargo list message</td>
</tr>
<tr>
<td>IFTMAN</td>
<td>Arrival notice message</td>
</tr>
<tr>
<td>IFTMBC</td>
<td>Booking confirmation message</td>
</tr>
<tr>
<td>IFTMBF</td>
<td>Firm booking message</td>
</tr>
<tr>
<td>IFTMBP</td>
<td>Provisional booking message</td>
</tr>
<tr>
<td>IFTMCA</td>
<td>Consignment advice message</td>
</tr>
<tr>
<td>IFTMCS</td>
<td>Instruction contract status message</td>
</tr>
<tr>
<td>IFTMIN</td>
<td>Instruction message</td>
</tr>
<tr>
<td>IFTRIN</td>
<td>Forwarding and transport rate information message</td>
</tr>
<tr>
<td>IFTSAI</td>
<td>Forwarding and transport schedule and availability information message</td>
</tr>
<tr>
<td>IFTSTA</td>
<td>International multimodal status report message</td>
</tr>
<tr>
<td>IFTSTQ</td>
<td>International multimodal status request message</td>
</tr>
<tr>
<td>IMPDEF</td>
<td>EDI implementation guide definition message</td>
</tr>
</tbody>
</table>
### Table 2  Batch Messages Defined in Version D00A  (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFENT</td>
<td>Enterprise accounting information message</td>
</tr>
<tr>
<td>INSDES</td>
<td>Instruction to despatch message</td>
</tr>
<tr>
<td>INSPRE</td>
<td>Insurance premium message</td>
</tr>
<tr>
<td>INSREQ</td>
<td>Inspection request message</td>
</tr>
<tr>
<td>INSRPT</td>
<td>Inspection report message</td>
</tr>
<tr>
<td>INVOIC</td>
<td>Invoice message</td>
</tr>
<tr>
<td>INVRPT</td>
<td>Inventory report message</td>
</tr>
<tr>
<td>IPPOAD</td>
<td>Insurance policy administration message</td>
</tr>
<tr>
<td>IPPOMO</td>
<td>Motor insurance policy message</td>
</tr>
<tr>
<td>ITRRPT</td>
<td>In transit report detail message</td>
</tr>
<tr>
<td>JAPRES</td>
<td>Job application result message</td>
</tr>
<tr>
<td>JINFEDE</td>
<td>Job information demand message</td>
</tr>
<tr>
<td>JOBAPP</td>
<td>Job application proposal message</td>
</tr>
<tr>
<td>JOBCON</td>
<td>Job order confirmation message</td>
</tr>
<tr>
<td>JOBMOD</td>
<td>Job order modification message</td>
</tr>
<tr>
<td>JOBOFF</td>
<td>Job order message</td>
</tr>
<tr>
<td>JUPREQ</td>
<td>Justified payment request message</td>
</tr>
<tr>
<td>LEDGER</td>
<td>Ledger message</td>
</tr>
<tr>
<td>LREACT</td>
<td>Life reinsurance activity message</td>
</tr>
<tr>
<td>LRECLM</td>
<td>Life reinsurance claims message</td>
</tr>
<tr>
<td>MEDPID</td>
<td>Person identification message</td>
</tr>
<tr>
<td>MEDPRE</td>
<td>Medical prescription message</td>
</tr>
<tr>
<td>MEDREQ</td>
<td>Medical service request message</td>
</tr>
<tr>
<td>MEDRPT</td>
<td>Medical service report message</td>
</tr>
<tr>
<td>MEDRUC</td>
<td>Medical resource usage and cost message</td>
</tr>
<tr>
<td>MEQPOS</td>
<td>Means of transport and equipment position message</td>
</tr>
<tr>
<td>MOVINS</td>
<td>Stowage instruction message</td>
</tr>
<tr>
<td>MSCONS</td>
<td>Metered services consumption report message</td>
</tr>
<tr>
<td>ORDCHG</td>
<td>Purchase order change request message</td>
</tr>
<tr>
<td>ORDERS</td>
<td>Purchase order message</td>
</tr>
<tr>
<td>ORDRSP</td>
<td>Purchase order response message</td>
</tr>
<tr>
<td>OSTENQ</td>
<td>Order status enquiry message</td>
</tr>
<tr>
<td>OSTRPT</td>
<td>Order status report message</td>
</tr>
<tr>
<td>PARTIN</td>
<td>Party information message</td>
</tr>
<tr>
<td>PAXLST</td>
<td>Passenger list message</td>
</tr>
</tbody>
</table>
### Table 2  Batch Messages Defined in Version D00A  (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYDUC</td>
<td>Payroll deductions advice message</td>
</tr>
<tr>
<td>PAYEXT</td>
<td>Extended payment order message</td>
</tr>
<tr>
<td>PAYMUL</td>
<td>Multiple payment order message</td>
</tr>
<tr>
<td>PAYORD</td>
<td>Payment order message</td>
</tr>
<tr>
<td>PRICAT</td>
<td>Price/sales catalogue message</td>
</tr>
<tr>
<td>PRIHIS</td>
<td>Pricing history message</td>
</tr>
<tr>
<td>PROCST</td>
<td>Project cost reporting message</td>
</tr>
<tr>
<td>PRODAT</td>
<td>Product data message</td>
</tr>
<tr>
<td>PRODEX</td>
<td>Product exchange reconciliation message</td>
</tr>
<tr>
<td>PROINQ</td>
<td>Product inquiry message</td>
</tr>
<tr>
<td>PROSRV</td>
<td>Product service message</td>
</tr>
<tr>
<td>PROTAP</td>
<td>Project tasks planning message</td>
</tr>
<tr>
<td>PRPAID</td>
<td>Insurance premium payment message</td>
</tr>
<tr>
<td>QALITY</td>
<td>Quality data message</td>
</tr>
<tr>
<td>QUOTES</td>
<td>Quote message</td>
</tr>
<tr>
<td>RDRMES</td>
<td>Raw data reporting message</td>
</tr>
<tr>
<td>REBORD</td>
<td>Reinsurance bordereau message</td>
</tr>
<tr>
<td>RECADV</td>
<td>Receiving advice message</td>
</tr>
<tr>
<td>RECALC</td>
<td>Reinsurance calculation message</td>
</tr>
<tr>
<td>RECECO</td>
<td>Credit risk cover message</td>
</tr>
<tr>
<td>RECLAM</td>
<td>Reinsurance claims message</td>
</tr>
<tr>
<td>RECORD</td>
<td>Reinsurance core data message</td>
</tr>
<tr>
<td>REGENT</td>
<td>Registration of enterprise message</td>
</tr>
<tr>
<td>RELIST</td>
<td>Reinsured objects list message</td>
</tr>
<tr>
<td>REMADV</td>
<td>Remittance advice message</td>
</tr>
<tr>
<td>REPREM</td>
<td>Reinsurance premium message</td>
</tr>
<tr>
<td>REQDOC</td>
<td>Request for document message</td>
</tr>
<tr>
<td>REQOTE</td>
<td>Request for quote message</td>
</tr>
<tr>
<td>RESETT</td>
<td>Reinsurance settlement message</td>
</tr>
<tr>
<td>RESMSG</td>
<td>Reservation message</td>
</tr>
<tr>
<td>RETACC</td>
<td>Reinsurance technical account message</td>
</tr>
<tr>
<td>RETANN</td>
<td>Announcement for returns message</td>
</tr>
<tr>
<td>RETINS</td>
<td>Instruction for returns message</td>
</tr>
<tr>
<td>RPCALL</td>
<td>Repair call message</td>
</tr>
<tr>
<td>SAFHAZ</td>
<td>Safety and hazard data message</td>
</tr>
</tbody>
</table>
2.1.3 **Segment Table**

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

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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANCRT</td>
<td>International movement of goods governmental regulatory message</td>
</tr>
<tr>
<td>SLSFCT</td>
<td>Sales forecast message</td>
</tr>
<tr>
<td>SLSRPT</td>
<td>Sales data report message</td>
</tr>
<tr>
<td>SOCade</td>
<td>Social administration message</td>
</tr>
<tr>
<td>SSIMOD</td>
<td>Modification of identity details message</td>
</tr>
<tr>
<td>SSRECH</td>
<td>Worker's insurance history message</td>
</tr>
<tr>
<td>SSREGW</td>
<td>Notification of registration of a worker message 1</td>
</tr>
<tr>
<td>STATAC</td>
<td>Statement of account message</td>
</tr>
<tr>
<td>STLRPT</td>
<td>Settlement transaction reporting message</td>
</tr>
<tr>
<td>SUPCOT</td>
<td>Superannuation contributions advice message</td>
</tr>
<tr>
<td>SUPMAN</td>
<td>Superannuation maintenance message</td>
</tr>
<tr>
<td>SUPRES</td>
<td>Supplier response message</td>
</tr>
<tr>
<td>TANSTA</td>
<td>Tank status report message</td>
</tr>
<tr>
<td>TAXCON</td>
<td>Tax control message</td>
</tr>
<tr>
<td>TPFREP</td>
<td>Terminal performance message</td>
</tr>
<tr>
<td>VATDEC</td>
<td>Value added tax message</td>
</tr>
<tr>
<td>VESDEP</td>
<td>Vessel departure message</td>
</tr>
<tr>
<td>WASDIS</td>
<td>Waste disposal information message</td>
</tr>
<tr>
<td>WKGRDC</td>
<td>Work grant decision message</td>
</tr>
<tr>
<td>WKGRRE</td>
<td>Work grant request message</td>
</tr>
</tbody>
</table>
4.3.1 Segment table

<table>
<thead>
<tr>
<th>Pos</th>
<th>Tag Name</th>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td>UNH Message header</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>0020</td>
<td>BGM Beginning of message</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>0030</td>
<td>DTM Date/time/period</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>0040</td>
<td>FTX Free text</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>0050</td>
<td>CNT Control total</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>0060</td>
<td>ÄÄÄÄÄ Segment group 1</td>
<td>ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ</td>
<td>9ÄÄÄÄÄÄÄÄÄÄÄ¿</td>
</tr>
<tr>
<td>0070</td>
<td>DOC Document/message details</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>0080</td>
<td>DTM Date/time/period</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>0090</td>
<td>ÄÄÄÄÄ Segment group 2</td>
<td>ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ</td>
<td>9ÄÄÄÄÄÄÄÄÄÄÄÙ</td>
</tr>
<tr>
<td>0100</td>
<td>RFF Reference</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>0110</td>
<td>DTM Date/time/period</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>0120</td>
<td>ÄÄÄÄÄ Segment group 3</td>
<td>ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ</td>
<td>9ÄÄÄÄÄÄÄÄÄÄÄ¿</td>
</tr>
<tr>
<td>0130</td>
<td>NAD Name and address</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>0140</td>
<td>CTA Contact information</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>0150</td>
<td>COM Communication contact</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>0160</td>
<td>ÄÄÄÄÄ Segment group 4</td>
<td>ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ</td>
<td>9ÄÄÄÄÄÄÄÄÄÄÄ¿</td>
</tr>
<tr>
<td>0170</td>
<td>ERC Application error information</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>0180</td>
<td>FTX Free text</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>0190</td>
<td>ÄÄÄÄÄ Segment group 5</td>
<td>ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ</td>
<td>9ÄÄÄÄÄÄÄÄÄÄÄ¿</td>
</tr>
<tr>
<td>0200</td>
<td>RFF Reference</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>0210</td>
<td>FTX Free text</td>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>0220</td>
<td>UNT Message trailer</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

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

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

2.1.4 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
2.1.5 Envelopes

UN/EDIFACT publishes the envelope segments in the separate syntax document with independent version numbers. For example, either syntax version 3 or syntax version 4 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>
<thead>
<tr>
<th>Table 3</th>
<th>Comparison Between X12 and UN/EDIFACT Envelopes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X12</strong></td>
<td><strong>ENVELOPE</strong></td>
</tr>
<tr>
<td>start</td>
<td>end</td>
</tr>
<tr>
<td>ISA</td>
<td>IEA</td>
</tr>
<tr>
<td>GS</td>
<td>GE</td>
</tr>
<tr>
<td>ST</td>
<td>SE</td>
</tr>
</tbody>
</table>

**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 OTD Library includes both a v3 **CONTRL** and a v4 **CONTRL** message. The user can select which one to use.
2.1.6 Delimiters

Delimiters are set dynamically.

2.1.7 OTD Libraries

The UN/EDIFACT OTD 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

If you need to make changes to a Java OTD, modify the SEF file, which is a text file, and regenerate.

To modify a SEF file

1 With the Java OTD Editor open, select File > New.
2 From the New Object 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 an 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 an 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 Object Type Definition and its associated Java classes.

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


---

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 4 on page 26. Table 5 on page 26 lists other related X12 and 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 (REMAADV 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 or UN/EDIFACT
Payment Order/Remittance Advice comparison, see Table 4. Table 5 lists other related X12 and UN/EDIFACT transactions.

**Table 4** Comparison of X12 to UN/EDIFACT: Payment Order/Remittance Advice

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

**Table 5** Other Related Transactions

<table>
<thead>
<tr>
<th>X12</th>
<th>Transaction</th>
<th>UN/EDIFACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>828</td>
<td>Debit Authorization</td>
<td>AUTHOR</td>
</tr>
<tr>
<td>829</td>
<td>Payment Cancellation Request</td>
<td>FINCAN</td>
</tr>
<tr>
<td>831</td>
<td>Application Control Totals</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### 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).
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 6, below, shows X12 and UN/EDIFACT headers.

**Table 6** Sample X12 and UN/EDIFACT Headers

<table>
<thead>
<tr>
<th>X12 start</th>
<th>X12 end</th>
<th>Envelope</th>
<th>UN/EDIFACT start</th>
<th>UN/EDIFACT end</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA</td>
<td>IEA</td>
<td>Interchange Envelope</td>
<td>UNA/UNB</td>
<td>UNZ</td>
</tr>
<tr>
<td>GS</td>
<td>GE</td>
<td>Functional Group Envelope</td>
<td>UNG</td>
<td>UNE</td>
</tr>
<tr>
<td>ST</td>
<td>SE</td>
<td>Message Envelope</td>
<td>UNH</td>
<td>UNT</td>
</tr>
</tbody>
</table>
Point-to-point scenario

In a sample 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 UNB) interchange header in which:
   - Sender ID = Company 1
   - Receiver ID = Bank 1

2. Bank 1 replaces Company 1’s ISA (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 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 CONTL) to the originator.

End-to-end scenario

In a sample 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 eXchange product automatically remove both the interchange and functional group envelopes and re-create the point-to-point envelopes. Special handling is required to override this default.

2.2.4 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 7.

<table>
<thead>
<tr>
<th>UN/EDIFACT</th>
<th>Envelope</th>
<th>X12</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTRL</td>
<td>System Level Acknowledgment (receipt)</td>
<td>TA1</td>
</tr>
<tr>
<td>CONTRL</td>
<td>Function Acknowledgment (point-to-point)</td>
<td>997</td>
</tr>
<tr>
<td></td>
<td>Application Advice (end-to-end)</td>
<td>824</td>
</tr>
</tbody>
</table>
2.3 Key Parts of EDI Processing Logic

The five key parts of EDI processing logic are listed in Table 8.

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

The UN/EDIFACT OTD 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.

2.3.1. Structures

The UN/EDIFACT OTD Library includes pre-built OTDs for all supported EDIFACT versions. These OTDs can be viewed in the OTDEditor, but cannot be modified.

Customization

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 SEFWizard to generate the OTD.

2.3.2. 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.

EDIFACT OTDs have validations and translations, but a validation does not generate an acknowledgment transaction. Instead, it generates a string.
Chapter 3

Installation

This chapter provides information on the installation procedure for the SeeBeyond UN/EDIFACT OTD Library and shows the resulting Project Explorer structure for the OTDs. It includes general installation information and installation instructions.

At this release, the UN/EDIFACT OTD Library includes templates for the versions shown in Table 9.

Table 9  UN/EDIFACT Versions Supported

<table>
<thead>
<tr>
<th>Version</th>
<th>Version</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.93A</td>
<td>D.97A</td>
<td>D.00A</td>
</tr>
<tr>
<td>D.95A</td>
<td>D.97B</td>
<td>D.00B</td>
</tr>
<tr>
<td>D.95B</td>
<td>D.98A</td>
<td>D.01A</td>
</tr>
<tr>
<td>D.96A</td>
<td>D.98B</td>
<td>D.01B</td>
</tr>
<tr>
<td>D.96B</td>
<td>D.99A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D.99B</td>
<td></td>
</tr>
</tbody>
</table>

Some additional points to note:

- Each product .sar file (UN_EDIFACT_OTD_Lib_v?_D##?.sar) requires from 10 MB to 20 MB disk space; thus, the combined disk space required to load all thirty .sar files (that is, v3 and v4 of D.93A through D.01B) is approximately 420 MB.

3.1  System Requirements

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

- Sun Solaris 8 and Solaris 9
- IBM AIX 5L Versions 5.1 and 5.2
- HP-UX 11.0, 11i (PA-RISC), and 11i V2.0 (11.23)
- HP Tru64 UNIX Version 5.1A
- Red Hat Linux 8 (Intel x86 version) and Advanced Server 2.1 (Intel x86 version)
3.2 Installation Procedure

The steps for installing the UN/EDIFACT OTD Library are the same as for other products in the ICAN Suite. You can find general product installation instructions in the *ICAN Suite Installation Guide*, which is available on the product media and can also be accessed via Enterprise Manager (Documentation tab).

Whenever a UN/EDIFACT OTD Library is installed, the Project Explorer tree adds a new project under the *SeeBeyond > OTD Library > EDIFACT* hierarchy. Each version of UN/EDIFACT, such as D.01B or D.99A, has a separate folder.

3.2.1. Uploading to the Repository

Before you begin

- A Repository server must be running on the machine where you will be uploading the product files.
- You must have already uploaded *eGate.sar*, and you must have already uploaded a *license.sar* file that includes a license for the UN/EDIFACT OTD library product.

To upload product files to the Repository

1. On a Windows machine, start a Web browser and point it at the machine and port where the Repository server is running:

   \[http://<hostname>:<port>\]

   where
   - `<hostname>` is the name of the machine running the Repository server.
   - `<port>` is the starting port number assigned when the Repository was installed.

   For example, the URL you enter might look like either of the following:

   http://localhost:12001
   http://serv1234.company.com:19876

2. In the Enterprise Manager *SeeBeyond Customer Login* page, enter your username and password.

3. When Enterprise Manager responds, click the *ADMIN* tab.

4. In the ADMIN page, click *Browse*.

5. In the *Choose file* dialog, click *ProductsManifest.xml*, and then click *Open*.

6. In the ADMIN page, click *Submit*.

   The lower half of the ADMIN page lists the product files you are licensed to upload.

7. In the Products column, find the UN EDIFACT OTD Library v# D##? product, and then click the *Browse* button for it.

8. In the *Choose file* dialog, click the corresponding *UN_EDIFACT_OTD_Lib[...].sar* file, and then click *Open*.

9. Repeat the previous two steps for other *.sar* files you want to upload, such as other OTD libraries, eXchange, or SME Web Services.

10. In the ADMIN page, click the *upload now :::* button.
3.2.2. **Refreshing Enterprise Designer**

**Before you begin**

- You must have already downloaded and installed Enterprise Designer, and a Repository server must be running on the machine where you uploaded the OTD Library product files.

**To refresh an existing installation of Enterprise Designer**

1. Start Enterprise Designer.

2. On the **Tools** menu, click **Update Center**.
   
   The Update Center shows a list of components ready for updating. See Figure 2.

**Figure 2  Update Center Wizard: Select Modules to Install**

3. Click **Add All** (the button with a doubled chevron pointing to the right).
   
   All modules move from the Available/New pane to the **Include in Install** pane.

4. Click **Next** and, in the next window, click **Accept** to accept the license agreement.

5. When the progress bars indicate the download has ended, click **Next**. Review the certificates and installed modules, and then click **Finish**. When prompted to restart Enterprise Designer, click **OK**.
   
   When Enterprise Designer restarts, the installation of the UN/EDIFACT OTD Library is complete, and you can use all library templates that you installed.
3.3 UN/EDIFACT Library Templates

3.3.1. UN/EDIFACT OTDs

Since there is an OTD for each EDIFACT transaction, installation of each version of EDIFACT includes a large number of OTDs. Figure 3 shows some of the OTDs installed for a specific version of EDIFACT (in this case, version D01B); compare with Table 2 on page 13.

**Figure 3** Some of the Transaction Set Structures for EDIFACT Version D00A

3.3.2. Transaction Template Names

The names for the EDIFACT templates are designed to assist you in quickly locating the file you want. The name for each transaction template is composed of the same set of elements in the same sequence. The names are constructed as follows:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>eDF</strong></td>
<td>abbreviated name of standard, followed by underscore</td>
</tr>
<tr>
<td><strong>v3</strong></td>
<td>syntax version, followed by underscore</td>
</tr>
<tr>
<td><strong>v4</strong></td>
<td></td>
</tr>
<tr>
<td><strong>D00A</strong></td>
<td>messaging version, followed by underscore</td>
</tr>
<tr>
<td><strong>APERAK</strong></td>
<td>six-character abbreviation for transaction name, followed by underscore</td>
</tr>
<tr>
<td><strong>_Full</strong></td>
<td>full transaction name</td>
</tr>
</tbody>
</table>
Examples:

- The name for an APERAK (Application error and acknowledgment message) in messaging version D00A, syntax version 4 is `eDF_v4_D00A_APERAK_AppErrorAndAcknMess_Full`
- The name for an OSTENQ (Order status enquiry message) in messaging version D01B, syntax version 3 is `eDF_v3_D01B_OSTENQ_ORDStatEnqMess_Full`

**eGate Project Explorer Display of UN/EDIFACT OTDs**

The UN/EDIFACT OTD Library is organized into a hierarchy as follows: In eGate Project Explorer, under the main `SeeBeyond` folder, the `OTD Library` folder holds all OTD Libraries you have installed, each in its own folder. Within the `EDIFACT` folder are two folders, `v3` and `v4`, corresponding to the two supported syntax versions. Both of these syntax-version folders has a folder for each library you have installed, such as D95A, D95B, D96A, ..., D01B. Each D##? folder contains nearly 200 OTDs, corresponding to the nearly 200 transactions that have been defined.

The Project Explorer hierarchy is shown in Table 10.

**Table 10  OTD Library Hierarchy in Project Explorer**

<table>
<thead>
<tr>
<th>Folder</th>
<th>Directory Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeeBeyond</td>
<td>Components built and supplied by SeeBeyond</td>
</tr>
<tr>
<td>OTD Library</td>
<td>Folders for each OTD library you have installed</td>
</tr>
<tr>
<td>EDIFACT</td>
<td>Folders specific to the EDIFACT OTD Library.</td>
</tr>
<tr>
<td>v3</td>
<td>Folders specific to EDIFACT syntax version v3</td>
</tr>
<tr>
<td>D...</td>
<td>OTDs for version D...</td>
</tr>
<tr>
<td>v4</td>
<td>Folders specific to EDIFACT syntax version v4</td>
</tr>
<tr>
<td>D95A</td>
<td>OTDs for version D95A</td>
</tr>
<tr>
<td>D...</td>
<td>OTDs for version D...</td>
</tr>
<tr>
<td>D01B</td>
<td>OTDs for version D01B</td>
</tr>
</tbody>
</table>

**Additional information**

For complete information on the purpose and function of each transaction, see the United Nations’ World Wide Web site that deals with the UN/EDIFACT standards. The URL for this Internet site is:

[http://www.unece.org/trade/untdid/welcome.htm](http://www.unece.org/trade/untdid/welcome.htm)

**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.
UN/EDIFACT OTD Library

This chapter lists sample file and directory names in the UN/EDIFACT OTD 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 OTD 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 4 on page 37
- Table 11 on page 37
- Table 12 on page 37
Note: These envelopes can be used with any version of the UN/EDIFACT OTD 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 4).

**Figure 4**  v3 Envelope Segments

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service String Advice</td>
<td>UNA</td>
<td>Conditional</td>
</tr>
<tr>
<td>Interchange Header</td>
<td>UNB</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Group Header</td>
<td>UNG</td>
<td>Conditional</td>
</tr>
<tr>
<td>Message Header</td>
<td>UNH</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Message Trailer</td>
<td>UNT</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Group Trailer</td>
<td>UNE</td>
<td>Conditional</td>
</tr>
<tr>
<td>Interchange Trailer</td>
<td>UNZ</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

**Table 11  v3 Control Message**

<table>
<thead>
<tr>
<th>v3 Control Message Name</th>
<th>v3 Control Message Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTRL</td>
<td>Syntactically acknowledges or rejects, with error indication, a received interchange, functional group, or message.</td>
</tr>
</tbody>
</table>

**Table 12  v3 Batch Segments**

<table>
<thead>
<tr>
<th>v3 Segment Name</th>
<th>v3 Segment Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCD</td>
<td>Data Element Error Indication</td>
</tr>
<tr>
<td>UCF</td>
<td>Functional Group Response</td>
</tr>
<tr>
<td>UCI</td>
<td>Interchange Response</td>
</tr>
<tr>
<td>UCM</td>
<td>Message Response</td>
</tr>
<tr>
<td>UCS</td>
<td>Segment Error Indication</td>
</tr>
<tr>
<td>UNA</td>
<td>Delimiter List</td>
</tr>
<tr>
<td>UNB</td>
<td>Interchange Header</td>
</tr>
<tr>
<td>UNE</td>
<td>Functional Group Trailer</td>
</tr>
<tr>
<td>UNG</td>
<td>Functional Group Header</td>
</tr>
<tr>
<td>UNH</td>
<td>Message Header</td>
</tr>
<tr>
<td>UNS</td>
<td>Section Control</td>
</tr>
</tbody>
</table>
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 5 on page 39
- Figure 6 on page 39
- Table 13 on page 40
- Table 14 on page 40

Note: These envelopes can be used with any version of the UN/EDIFACT OTD 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 5).
Figure 5  v4 Batch Envelope Segments

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service String Advice</td>
<td>UNA</td>
<td>Conditional</td>
</tr>
<tr>
<td>Interchange Header</td>
<td>UNB</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Group Header</td>
<td>UNG</td>
<td>Conditional</td>
</tr>
<tr>
<td>Message Header</td>
<td>UNH</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

Message Body Segments

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Trailer</td>
<td>UNT</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Group Trailer</td>
<td>UNE</td>
<td>Conditional</td>
</tr>
<tr>
<td>Interchange Trailer</td>
<td>UNZ</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

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 6).

Figure 6  v4 Interactive Envelope Segments

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service String Advice</td>
<td>UNA</td>
<td>Conditional</td>
</tr>
<tr>
<td>Interactive Interchange Header</td>
<td>UIB</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Interactive Message Header</td>
<td>UIH</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

Message Body Segments

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive Message Trailer</td>
<td>UIT</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Interactive Interchange Trailer</td>
<td>UIZ</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>
Table 13  v4 Control Message

<table>
<thead>
<tr>
<th>v4 Control Message Name</th>
<th>v4 Control Message Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTACK</td>
<td>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.</td>
</tr>
<tr>
<td>CONTRL (for batch EDI)</td>
<td>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.</td>
</tr>
<tr>
<td>KEYMAN</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 14  v4 Segments

<table>
<thead>
<tr>
<th>v4 Segment Name</th>
<th>v4 Segment Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCD</td>
<td>Data Element Error Indication</td>
</tr>
<tr>
<td>UCF</td>
<td>Group Response</td>
</tr>
<tr>
<td>UCI</td>
<td>Interchange Response</td>
</tr>
<tr>
<td>UCM</td>
<td>Message / Package Response</td>
</tr>
<tr>
<td>UCS</td>
<td>Segment Error Indication</td>
</tr>
<tr>
<td>UGH</td>
<td>Anti-Collision Segment Group Header</td>
</tr>
<tr>
<td>UGT</td>
<td>Anti-Collision Segment Group Trailer</td>
</tr>
<tr>
<td>UIB</td>
<td>Interactive Interchange Header</td>
</tr>
<tr>
<td>UIH</td>
<td>Interactive Message Header</td>
</tr>
<tr>
<td>UIR</td>
<td>Interactive Status</td>
</tr>
<tr>
<td>UIT</td>
<td>Interactive Message Trailer</td>
</tr>
</tbody>
</table>
### Table 14  v4 Segments (Continued)

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

**Note:** UNA is optional and seldom used.
Chapter 5

Working With the EDIFACT OTDs

This chapter provides information on additional features built into the EDIFACT OTDs, and instructions on working with the OTDs and on testing them.

• See “Viewing an EDIFACT OTD in the OTD Editor” on page 43.

It also provides information on using the Java methods provided within the OTDs, and other general information about using the UN/EDIFACT OTD Library.

• See “Setting the Delimiters” on page 46 and “Methods for Getting and Setting” on page 46. (Further details are provided in Chapter 5 “Java Methods for X12 OTDs” on page 52.)

To test that your data is being mapped correctly by the OTD, and that the data is valid based on definitions and business rules, you can run `performValidation()` within the Java Collaboration Editor.

• See “Using Validation in the Java Collaboration Editor” on page 48.

Information on limitations you should know about the X12 OTD Library is provided in “Limitations of EDIFACT OTDs” on page 52.

5.1 Importing .jar Files

If your project contains one or more Java collaborations that access bean nodes for reporting errors and exceptions of EDIFACT OTDs (see “Bean Nodes for Reporting Errors and Exceptions” on page 53), then you must import a .jar file as described below.

Before you begin

• You must have completed all the other installation steps.

• A Repository server must be running on the machine where you uploaded the EDIFACT OTD Library product files.

• You must have already created a project.

To import com.stc.otd.sefimpl.jar

1. Start Enterprise Designer and, if necessary, create a project.

2. Right-click the project and, on the popup context menu, point at New and click File.
3 Navigate to the folder `<ican50>\edesigner\usrdir\modules\ext\sefwizard\`, select `com.stc.otd.sefimpl.jar`, and click **Import**. See Figure 7.

**Figure 7** Importing sefimpl.jar

![Image of Importing sefimpl.jar]

4 Later, in the Java Collaboration Editor, you will use the **Import JAR file** button on the tool palette to add `sefimpl.jar` to a collaboration that uses the X12 OTD.

5.2 **Viewing an EDIFACT OTD in the OTD Editor**

To view an EDIFACT OTD (or any other OTD), simply double-click the name in the Project Explorer tree. The OTD Editor automatically opens to display it. Within the OTD Editor, you can expand or contract a parent node by single-clicking the icon to its left, or by double-clicking the node name. For some of the items, help is available by hovering your cursor over the item.

For elements other than bean nodes, the following naming conventions apply:

- Each element name begins with `E`
- Each segment loop name begins with `Loop`

An example of an X12 270 transaction in the OTD Editor is shown in Table 8 on page 45. The OTD shown in Figure 8 is `x12_4010_270_EligCoveOrBeneInqu_Full`. Some of its parent nodes are fully expanded, some partly so, and some full collapsed. In this example, the root node is `X12_4010_270_EligCoveOrBeneInqu_Outer`. This pattern holds for all the X12 OTDs: the root node name is the same as the OTD name, but with the first letter in upper case (X instead of x) and the string `_Outer` replacing the string.
_Full. Under this root node, the first node is the ISA header node, and then comes the node X12_4010_270_EligCoveOrBeneInqu_Inner, which references the enveloping information.
Figure 8  X12 270 Transaction in the OTD Editor
5.3 Setting the Delimiters

The OTDs must include some way for delimiters to be defined so that they can be mapped successfully from one OTD to another. The X12 delimiters are as follows:

- Data Element Separator (default is an asterisk)
- Subelement Separator / Component Element Separator (default is a colon)
- Repetition Separator (version 4020 and later) (default is a plus sign)
- Segment Terminator (default is a tilde)

Two delimiters—Repetition Separator and Subelement Separator—are explicitly specified in the interchange header segment (ISA). The other two delimiters are implicitly defined within the structure of the ISA, by their first usage. For example, after the fourth character defines the Data Element Separator, the same character is used subsequently to delimit all data elements; and after the 107th character defines the Segment Terminator, the same character is used subsequently to delimit all segments.

Because the OTD automatically detects delimiters while unmarshaling, you need not (and should not) specify delimiters for an incoming message; any delimiters that are set before unmarshaling are ignored, and the unmarshal() function picks up the delimiter being used in the ISA segment of the incoming message.

You can specify delimiters in two ways:

- You can set the Subelement Separator and Repetition Separator from the corresponding elements within the ISA segment.
- You can set the delimiters in the Java Collaboration Editor using bean nodes that are provided in the OTDs. Specific information on using bean nodes to get and set these delimiter values is provided in Chapter 5:

  - elementSeparator (see `getElementSeparator` on page 58)
  - subelementSeparator (see `getSubelementSeparator` on page 62)
  - repetitionSeparator (see `getRepetitionSeparator` on page 60)
  - segmentTerminator (see `getSegmentTerminator` on page 61)

If the input data is already in X12 format, you can use the “get” methods to get the delimiters from the input data. If the collaboration is putting the data into X12 format, you can use the “set” methods to set the delimiters in the output OTD. See “Methods for Getting and Setting” on page 46.

5.4 Methods for Getting and Setting

Bean nodes automatically have get and set methods associated with them; in other words, a bean node named theBeanNode has a method `getTheBeanNode()` to read the current value and another method `setTheBeanNode()` to write a value. Therefore, do not assume that a node is read/write merely because it has a `setNode()` method.
5.4.1. **Bean Nodes for Getting and Setting Data**

The following bean nodes are available under the root node and at the xxx_Outer, xxx_Inner, and xxx (transaction set) levels:

- `elementSeparator(char)` — to get or set the element separator.
- `inputSource(byte[])` — to get the byte array of original input data source.
- `repetitionSeparator(char)` — to get or set the repetition separator.
- `segmentCount(int)` — to get the segment count at the current level. This node is also available for segment loops.
- `segmentTerminator(char)` — to get or set the segment terminator.
- `subelementSeparator(char)` — to get or set the subelement separator.
- `xmlOutput(boolean)` — to set whether the output should be in XML format.

The following bean node is available from the Loop elements:

- `segmentCount(int)` — to get the segment count at the current level. This node is also available under the root node and at the xxx_Outer, xxx_Inner, and xxx (transaction set) levels.

5.4.2. **Bean Nodes for Getting Errors and Results**

The following bean nodes are available under the root node and at the xxx_Outer, xxx_Inner, and xxx (transaction set) levels.

- `allErrors(String[])` — to get errors during unmarshaling from the input data and validation results on message and envelopes, in the format of a String array that combines (without duplication) the results from ICValidationResult(), FGValidationResult(), TSValidationResult(), and msgValidationResult().
- `ICValidationResult(com.stc.otd.runtime.check.sef.ICError[])` — to get the interchange envelope validation result, in the format of an array of com.stc.otd.runtime.check.sef.ICError objects.
- `FGValidationResult(com.stc.otd.runtime.check.sef.FGError[])` — to get the functional group envelope validation result in the format of an array of com.stc.otd.runtime.check.sef.FGError objects.
- `TSValidationResult(com.stc.otd.runtime.check.sef.TSError[])` — to get the transaction set envelope validation result in the format of an array of com.stc.otd.runtime.check.sef.TSError objects.
- `maxDataError(int)` — to get or set the maximum number of validation errors to be reported, where -1 means “no limit.”
- `msgValidationResult(com.stc.otd.runtime.check.sef.DataError[])` — to get validation errors, in the format of com.stc.otd.runtime.check.sef.DataError objects.
- `unmarshalErrors(com.stc.otd.runtime.check.sef.DataError[])` — to get errors that occurred during unmarshaling from the input data, in the format of
com.stc.otd.runtime.check.sef.DataError objects. The presence of any objects in this array implies that isUnmarshalComplete() is false.

5.5 Using Validation in the Java Collaboration Editor

Each of the OTDs in the X12 OTD library includes a Java method for the purpose of validating your data:

- performValidation()

Information on using this method from within Java Collaboration Editor (JCE) GUI is provided below. Technical information on the Java methods is provided in "Java Methods for X12 OTDs" on page 52.

5.5.1. Creating a Collaboration Rule to Validate an OTD

The elements that are part of an OTD can be dragged and dropped when two or more OTDs are opened in the Java Collaboration Editor; see the eGate Integrator User’s Guide for more information. A field on the input (left) side pane can be dragged to a field in the output (right) pane. This action, when highlighted in the Business Rules pane, displays the rule in the Rule Properties pane.

To access the method, right-click the node and, on the context popup menu, click **Select a method to call**. See Figure 9.

![Figure 9 Accessing a Method in an X12 OTD](image)

The methods available depend on the node you select. In particular, if you right-click the root node of the OTD, one of the methods available to you is **performValidation()**; see Figure 10.
5.6 Alternative Formats: ANSI and XML

All the EDIFACT OTDs accept either standard ANSI X12 format or XML format as input, by default; and, by default, output from a collaboration that uses messages from an EDIFACT OTD is in UN/EDIFACT format. However, there is a Java method available for setting the output to XML:

- setXMLOutput (boolean isXML)

If you want to set the collaboration to output XML format, use setXmlOutput(true); in other words, set the xmlOutput bean node to the value true.

5.6.1. XML Format for EDIFACT

Since there is no de facto XML standard for EDIFACT as yet, the SeeBeyond EDIFACT OTD Library uses Open Business Objects for EDI (OBOE) as the XML format for EDIFACT.

The XML EDIFACT DTD is shown in Figure 11.
Figure 11  XML EDIFACT DTD

```xml
<!ELEMENT envelope (segment, segment?, functionalgroup+, segment)>
<!ATTLIST envelope format CDATA #IMPLIED>

<!ELEMENT functionalgroup (segment, transactionset+, segment)>
<!ELEMENT transactionset (table+)>
<!ATTLIST transactionset code CDATA #REQUIRED>
<!ATTLIST transactionset name CDATA #IMPLIED>

<!ELEMENT table (segment)+>
<!ATTLIST table section CDATA #IMPLIED>

<!ELEMENT segment ((element | composite)+, segment*)>
<!ATTLIST segment code CDATA #REQUIRED>
<!ATTLIST segment name CDATA #IMPLIED>

<!ELEMENT composite (element)+>
<!ATTLIST composite code CDATA #REQUIRED>
<!ATTLIST composite name CDATA #IMPLIED>

<!ELEMENT element (value)>
<!ATTLIST element code CDATA #REQUIRED>
<!ATTLIST element name CDATA #IMPLIED>

<!ELEMENT value (#PCDATA)>
<!ATTLIST value description CDATA #IMPLIED>
```
Figure 12 shows an EDIFACT 997 Functional Acknowledgment, in XML format.

**Figure 12** EDIFACT 997 Functional Acknowledgment—XML

An example of the same transaction, an EDIFACT 997 Functional Acknowledgment, using standard ANSI format, is shown in Figure 13.

**Figure 13** EDIFACT 997 Functional Acknowledgment—ANSI Format
5.7 Possible Differences in Output When Using Pass-Through

If you are using pass-through, the output file contains essentially the same data as the input file.

Certain differences in output, based on variations in acceptable interpretation of the information, are acceptable, provided that the data conforms to the formats specified for the elements. For example:

- If the input file includes a six-digit date, the output file might represent this as an eight-digit value. For example, 040715 in the input file might be represented as 20040705 in the output file.
- The number of trailing zeros after a decimal point might vary. For example, an input value of 10.000 might be represented as 10 in the output file.

The reason these changes occur is that, during pass-through, certain data fields are parsed and stored as Java objects other than strings; for example, Date or Double. The actual value of all the information must remain the same.

5.8 Limitations of EDIFACT OTDs

5.8.1. Memory Requirements

When using an EDIFACT OTD, set the maximum heap size to more than 128MB for Enterprise Designer and the OTD tester; a value of 256MB is recommended. If settings are 128MB or less, and multiple messages are processed simultaneously (such as during FTP batch upload of messages), a NullPointerException can occur.

- To set the heap size in Enterprise Designer: On the Tools menu, click Options.

**Figure 14** Setting the Maximum Heap Size
5.8.2. **Delayed Unmarshaling**

For performance reasons, the `unmarshal()` method does not unmarshal the entire message. Instead, it does the following:

- Unmarshals the incoming message at the segment level. In other words, the OTD checks for all relevant segments and reports any missing or extra segments.
- Reports trailing delimiter for elements and composites.

Elements within a segment are not unmarshaled until an element in that segment is accessed in the collaboration using a `getXxx()` method.

5.8.3. **Errors and Exceptions**

For all EDIFACT OTDs, including the two `envelope` OTDs, if the incoming message cannot be parsed (for example, if the OTD cannot find the ISA segment), then the `unmarshal()` method throws a `com.stc.otd.runtime.UnmarshalException`.

You can also use the `isUnmarshalComplete()` method to learn whether `unmarshal()` executed without reporting any errors. Successful completion does not guarantee that the OTD instance is free of unmarshal exceptions within segments, however, since elements are not unmarshaled until the first `getElementXxx()` method of a segment is encountered (see “Delayed Unmarshaling” on page 53). Encountering this triggers an automatic background unmarshal of the entire segment, and any problems with the segment will be appended in `allErrors`, `unmarshalErrors`, and `msgValidationResult`. Note that the value returned by `isUnmarshalComplete()` is not influenced by the outcome of the automatic background unmarshal; instead, its value reflects what was set by the explicit invocation of the `unmarshal()` method.

It is an error to use the `setXxx` method if bean node `Xxx` is read-only. The system may throw compile exceptions if this is attempted.

If trailing element separators are found inside a segment

In the initial unmarshaling process, the OTD tries to parse the message based on the segment sequence defined in the input metadata (SEF) file. At the same time, however, it also checks for the presence of one or more trailing element separators inside the segment. If found, error arrays such as for `unmarshalErrors` are populated accordingly. Trailing element separators do not affect data parsing. Immediately after `unmarshal()` is invoked, therefore, if `isUnmarshalComplete()` returns `true`, then the error arrays for `unmarshalErrors` either contain no entries or else contain only errors of trailing element separators; if it returns `false`, the error arrays contain errors other than those of trailing element separators.

See “Bean Nodes for Reporting Errors and Exceptions” on page 53.
5.8.4. Special Methods for Error Classes

The `toString()` and `marshal()` methods of the following error classes cannot be implemented via the GUI, and have to be hand-coded:

- `com.stc.otd.runtime.check.sef.DataError`
- `com.stc.otd.runtime.check.sef.TSError`
- `com.stc.otd.runtime.check.sef.FGError`
- `com.stc.otd.runtime.check.sef.ICError`

Also see “Bean Nodes for Reporting Errors and Exceptions“ on page 53.
Chapter 6

Java Methods for EDIFACT OTDs

This chapter lists and explains the bean nodes and Java methods that are used to extend the functionality of the OTDs in the UN/EDIFACT OTD Library.

Note: For detailed information about the OTD Editor and the Java Collaboration Editor, see the eGate Integrator User’s Guide.

6.1 Using the OTD Editor to View and Test an OTD

The OTD Editor allows you to load, view, and test any OTD. You cannot edit UN/EDIFACT OTDs; however, you can view the OTD structure and validate it against sample data using the OTD Tester.

In an UN/EDIFACT OTD, all elements begin with an “E.” For example:

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

Note: If an item in an OTD starts with a “Loop”, it is a segment loop (for example: LoopRFF_5_Refe).

To open an OTD

1. In the OTD 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 15.

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.
6.1.1 Delimiters

UN/EDIFACT uses six delimiters. These delimiters, which the user can get and set using Java methods, 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 example shown in Figure 15. For example: `setDefaultEdifactDelimiters`.

### 6.1.2 Using the Collaboration Rules Editor to Validate an OTD

The elements that are part of an OTD can be dragged and dropped when two or more OTDs are opened in the Collaboration Rules Editor (see the eGate 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.

### 6.2 Methods

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

#### 6.2.1 Java Methods to Set or Get Delimiters

Each %%% (such as `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 OTD Template Library.

The `com.stc.edifact_v3_d95B.EDF...Outer` class extends `com.stc.jcsre.EDFOTDImpl` and implements `com.stc.jcsre.OTD`.

The `com.stc.edifact_v3_d95B.EDF...Outer` methods are:

- `setDefaultEdifactDelimiters` on page 58
- `getSegmentTerminator` on page 58
- `setSegmentTerminator` on page 59
- `getElementSeparator` on page 60
- `setElementSeparator` on page 60
- `getSubelementSeparator` on page 61
- `setSubelementSeparator` on page 61
- `getRepetitionSeparator` on page 62
- `setRepetitionSeparator` on page 63
The UN/EDIFACT OTD Library also includes the following custom method for testing the validation Collaboration:

- `validate` on page 63

---

**setDefaultEdifactDelimiters**

**Description**
Sets the current delimiters to the default UN/EDIFACT delimiters:

- `segmentTerminator = '`
- `elementSeparator = +`
- `subelementSeparator = :`
- `repetitionSeparator = *`

**Syntax**
```
public void setDefaultEdifactDelimiters()
```

**Parameters**
None

**setDefaultEdifactDelimiters Constants**
None

**Returns**
Void

**Throws**
None

**Example**
```
com.stc.edifact_v3_d95B.EDF_..._...Outer myOTD=new com.stc.edifact_v3_d95B.EDF_..._...Outer();
.......
myOTD.setDefaultEdifactDelimiters();
```

---

**getSegmentTerminator**

**Description**
Gets the segmentTerminator character.

**Syntax**
```
public char getSegmentTerminator()
```

**Parameters**
None
getSegmentTerminator Constants
   None

Returns
   char
   Returns the segment terminator character.

Throws
   None

Example
   com.stc.edifact_v3_d95B.EDF_..._...Outer myOTD=new com.stc.edifact_v3_d95B.
   EDF_..._...Outer();
   ......
   ......
   char segTerm=myOTD.getSegmentTerminator();

setSegmentTerminator

Description
   Sets the segmentTerminator character.

Syntax
   public void setSegmentTerminator(char c)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>char</td>
<td>The segmentTerminator character</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to be set.</td>
</tr>
</tbody>
</table>

setSegmentTerminator Constants
   None

Returns
   Void

Throws
   None

Example
   com.stc.edifact_v3_d95B.EDF_..._...Outer myOTD=new com.stc.edifact_v3_d95B.
   EDF_..._...Outer();
   ......
   ......
   ......
   char c='~';
   myOTD.setSegmentTerminator(c);
getElementSeparator

Description
Gets the elementSeparator character.

Syntax
public char getElementSeparator()

Parameters
None

generateElementSeparator Constants
None

Returns
char
Returns the element separator character.

Throws
None

Example
com.stc.edifact_v3_d95B.EDF_..._....Outer myOTD=new com.stc.edifact_v3_d95B.
EDF_....Outer();
......
......
char elmSep=myOTD.getElementSeparator();

setElementSeparator

Description
Sets the elementSeparator character.

Syntax
public void setElementSeparator(char c);

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>char</td>
<td>The elementSeparator character to be set.</td>
</tr>
</tbody>
</table>

setElementSeparator Constants
None

Returns
Void
Throws

None

Examples

com.stc.edifact_v3_d95B.EDF....Outer myOTD=new com.stc.edifact_v3_d95B.EDF....Outer();
......
......
char c='+';
myOTD.setElementSeparator(c);

getSubelementSeparator

Description

Gets the subelementSeparator character.

Syntax

public char getSubelementSeparator()

Parameters

None

getSubelementSeparator Constants

None

Returns

char

Returns the getSubelement character.

Throws

None

Example

com.stc.edifact_v3_d95B.EDF....Outer myOTD=new com.stc.edifact_v3_d95B.EDF....Outer();
......
......
char subeleSep=myOTD.getSubelementSeparator();

setSubelementSeparator

Description

Sets the SubelementSeparator character.

Syntax

public void setSubelementSeparator(char c)
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>char</td>
<td>The SubelementSeparator character to be set.</td>
</tr>
</tbody>
</table>

**setSubelementSeparator**

- **Constants**
  - None
- **Returns**
  - Void
- **Throws**
  - None
- **Example**
  ```java
class com.stc.edifact_v3_d95B.EDF_..._...Outer
  
  public static void main(String[] args)
  
  {
    com.stc.edifact_v3_d95B.EDF_..._...Outer myOTD=
    new com.stc.edifact_v3_d95B.EDF_..._...Outer();
    .......... 
    char c=':';
    myOTD.setSubelementSeparator(c);
  }
```

**getRepetitionSeparator**

- **Description**
  - Gets the RepetitionSeparator character.
- **Syntax**
  ```java
  public char getRepetitionSeparator()
  
  Parameters
  - None
  
  **getRepetitionSeparator**
  - **Constants**
    - None
  - **Returns**
    - char
    - Returns the getRepetitionSeparator character.
  - **Throws**
    - None
  - **Examples**
    ```java
    class com.stc.edifact_v3_d95B.EDF_..._...Outer
    
    public static void main(String[] args)
    
    {
      com.stc.edifact_v3_d95B.EDF_..._...Outer myOTD=
      new com.stc.edifact_v3_d95B.EDF_..._...Outer();
      .......... 
      char repSep=myOTD.getRepetitionSeparator();
    }
    ```
setRepetitionSeparator

Description
Sets the RepetitionSeparator character.

Syntax
public void setRepetitionSeparator(char c)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>char</td>
<td>The RepetitionSeparator character to be set.</td>
</tr>
</tbody>
</table>

setRepetitionSeparator Constants
None

Returns
Void

Throws
None

Example
com.stc.edifact_v3_d95B.EDF_..._...Outer myOTD=new com.stc.edifact_v3_d95B.EDF_..._...Outer();
......
......
char c='*';
myOTD.setRepetitionSeparator(c);

validate

Description
Invoking validate with no parameters validates the OTD content in memory.

Invoking validate with a single parameter, which must be of type boolean, validates the OTD content, either immediately after unmarshaling or in memory. When the value of the parameter is false, this method works exactly as validate with no parameters. When the value of the parameter is true, this method can be used to validate length information in the input data file.

Syntax (no parameters)
public java.lang.String validate()

Syntax (one parameter)
public String validate(boolean original)
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>boolean</td>
<td>If true, validates the OTD content right after unmarshaling. If false, validates the OTD in memory.</td>
</tr>
</tbody>
</table>

validate Constants

None.

Returns

java.lang.String

A description of the errors found in the data. If there are no errors, the string is null.

Throws

None.

Examples

```java
com.stc.edifact_v3_d95B.EDF.....Outer myOTD=new com.stc.edifact_v3_d95B.EDF.....Outer();
......
string msg=myOTD.validate();

com.stc.edifact_v3_d95B.EDF.....Outer myOTD=new com.stc.edifact_v3_d95B.EDF.....Outer();
......
......
string msg=myOTD.validate(true);```
Index

A
acknowledgments
as part of EDI logic 30
payment 29

B
batch messages
defined in Version D00A 13
bean nodes 46
allErrors 47
elementSeparator 46, 47
FGValidationResult 47
ICVValidationResult 47
inputSource 47
maxDataError 47
msgValidationResult 47
repetitionSeparator 46, 47
segmentCount 47
segmentTerminator 46, 47
subelementSeparator 46, 47
TSValidationResult 47
unmarshalErrors 47
xmlOutput 47

c
com.stc.otd.sefimpl.jar
(illustrated) 43
importing 43
compatible systems 8
Component Element Separator 46
components
of UN/EDIFACT 11
control messages 20
v4 40
conventions
path name separator 10
Windows 9

D
Data Element Separator 46
DataError 54
delimiters 12, 46, 56
Component Element Separator 46
Data Element Separator 46
Repetition Separator 46
Segment Terminator 46
Subelement Separator 46
document
conventions 9
organization 9

E
EDI
payment processing
exchange of payment orders 23
exchange of remittance information 23
formats for transporting a payment 24
functions a payment must perform 24
issuance of payment order 24
overview 22
routing of remittance information 23
processing
key terms 30
EDISIM 30
element separators
trailing 53
elementSeparator 46
dend-to-end scenario 28
envelopes 20
enveloping
as part of EDI logic 30
enveloping scenarios
end-to-end 28
point-to-point 28
understanding 26
error arrays
and unmarshalErrors() 47, 53
error classes
hand-coding 54
Event Type Definition
message structure in EDI 12

F
FGError 54
files
SEF 12
Foresight Corporation 30

G
getAllErrors 47
getElementSeparator 47
Index

getFGValidationResult 47
getICValidationResult 47
getInputSource 47
getMaxDataError 47
getMsgValidationResult 47
getRepetitionSeparator 47
getSegmentCount 47
getSegmentTerminator 47
getSubelementSeparator 47
getTSValidationResult 47
getUnmarshalErrors 47

H
hierarchy in Project Explorer
  UN/EDIFACT OTDs 35

I
ICError 54
  implementation 30
  installation 32–34
  isUnmarshalComplete 53

J
Java methods 57
  getAllErrors 47
  getElementSeparator 47
  getFGValidationResult 47
  getICValidationResult 47
  getInputSource 47
  getMsgValidationResult 47
  getRepetitionSeparator 47
  getSegmentCount 47
  getSegmentTerminator 47
  getSubelementSeparator 47
  getTSValidationResult 47
  getUnmarshalErrors 47
  isUnmarshalComplete 53
  marshal 54
  performValidation 48
  setElementSeparator 47
  setInputSource 47
  setMaxDataError 47
  setRepetitionSeparator 47
  setSegmentCount 47
  setSegmentTerminator 47
  setSubelementSeparator 47
  setXmlOutput 47
  toString 54
  unmarshal 53

K
key terms
  EDI processing 30

L
library templates 34
  loops 19

M
marshal
  limitations of 54
message names
  size 36
message structure
  defined 12
  OTD in eGate 12
methods
  for getting values 46
  for setting values 46

O
OTD names 34
  OTDs, working with 42–54

P
payment acknowledgment 29
  Payment Order
    comparison between X12 and UN/EDIFACT 26
  performValidation 48
  point-to-point scenario 28

R
reader 8
  Remittance Advice
    comparison between X12 and UN/EDIFACT 26
  Repetition Separator 46
  repetitionSeparator 46
  runtime exceptions
    UnmarshalException 53

S
SEF file 30
  sefimpl.jar
    (illustrated) 43
    adding to Collaboration 43
    importing 42
  SEFWizard 30
segment names
  size 36
segment table
  example of 18
Segment Terminator 46
segmentTerminator 46
setElementSeparator 47
setInputSource 47
setMaxDataError 47
setRepetitionSeparator 47
setSegmentCount 47
setSegmentTerminator 47
setSubelementSeparator 47
setXmlOutput 47, 49
structures 30
  as part of EDI logic 30
Subelement Separator 46
subelementSeparator 46
supporting documents 10
system requirements 31

T

template installation 32–34
toString
  limitations of 54
toString!
  limitations of 54
trailing element separators 53
transaction set structures 34
translations
  as part of EDI logic 30
TSError 54

U

UN/EDIFACT
  compared with X12 22
    Payment Order/Remittance Advice 26
  components of 11
    envelopes
      compared to X12 20
    overview 11
  point-to-point example 25
  types of acknowledgments
    compared to X12 29
United Nations URL
  for additional information 18
UN/EDIFACT ETD Library 21
directories
  overview of 36
files
  overview of 36
UN/EDIFACT OTD Library

v3
  batch control messages 37
  batch segments 37
  required for every EDIFACT message 36
v4
  control messages 40
  required for every EDIFACT message 36
  segments 40
  validations
    as part of EDI logic 30
  Version D00A 13

W

working with OTDs 42–54
writing conventions 9

X

X12
  compared with UN/EDIFACT 22
  EDIFACT comparison
    of Payment Order/Remittance Advice 26
  end-to-end example 25
    envelopes
      compared to UN/EDIFACT 20
    OTD names 34
  types of acknowledgments
    compared to UN/EDIFACT 29
X12 library templates 34
X12 template installation 32–34