IBM WebSphere Transformation Extender

Pack for EDIFACT

Version 2.7

Note
Before using this information, be sure to read the general information in "Notices" on page 13.

30 June 2006

This edition of this document applies to IBM WebSphere Transformation Extender Pack for EDIFACT Version 2.7; and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email DTX_doc_feedback@us.ibm.com . We look forward to hearing from you.

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

Contents

Chapter 1. Introduction 1
Installation 1

Chapter 2. EDIFACT type trees 3
Overview 3
EDIFACT 3
Interchange 3
Transmission 3
EDIFACT category 4
Control sub-tree 4
Group sub-tree 6
Msg sub-tree 6
Version specific sub-tree 6
Interchange category 8
Restart attribute on functional group 8
Component rule on UNZ segment 8
Transmission category 9
Restart attribute on an interchange 9
Mapping EDIFACT data 9
Input card - EDIFACT data from multiple trading partners 10
Output card - sending EDIFACT data to multiple partners 10

Chapter 3. Example files 11
EDIFACT example files 11
EDIFACT standard type trees 11

Notices 13
Programming interface information 15
Trademarks and service marks 15

Index 17

Chapter 1. Introduction

xThe Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) type trees are used to validate and ultimately transform, data that conforms to the United Nation's (UN's) EDIFACT standard, as described in the UN's EDIFACT specifications.

The EDIFACT standard is mainly prevalent in Europe. The EDIFACT standard is maintained by the UN and the specifications can be obtained from the www.unece.org/trade/untdid/welcome.htm website, with links for the Industry specific sub-standards.

Installation

These trees are available to download from the IBM ESD site: www-306.ibm.com/software/

Note: You will need your **User ID** and **Password** to login.

Chapter 2. EDIFACT type trees

The type trees described in this documentation can be found in the IBM WebSphere Transformation Extender Pack for EDIFACT.

Overview

The root type of each EDIFACT type tree is **EDI**. Each type tree has **EDIFACT**, **Interchange**, and **Transmission** categories stemming from the **EDI** root type. For example, the **edif91_1.mtt** type tree contains the EDIFACT version 91.1 and the **edif93_2.mtt** type tree contains the EDIFACT version 93.2.

EDIFACT trees are organized in the same way that ANSI X12 trees are organized. If you trade using both ANSI X12 and EDIFACT, you can easily design a type tree to make it a multi-standard tree.

EDIFACT

The following are subtypes of the **EDIFACT** category:

- **Control** type defines the EDIFACT control structure objects. The **Control** category is the same on each **EDIFACT** version type tree.
- **Group** type defines the functional groups for the version.
- **Msg** type defines the messages for the version.
- **V** category defines the message contents, segments, composites and elements in that particular version.

The version-specific data objects can be found under the category whose name begins with a **V** followed by the version number. For example, **V93_2** is the name of the category in the **edif93_2.mtt** version type tree.

For further information on these subtypes, see the ″EDIFACT Category″ section.

Interchange

The **Interchange** type contains the data types for an interchange.

The **Interchange** category contains **Inbound** and **Outbound** partitioned group subtypes.

Transmission

The **Transmission** type contains the data types for a transmission. A data object is made up of one or more interchanges.

The **Transmission** category contains **Inbound** and **Outbound** group subtypes.

Note: The EDIFACT type trees include both inbound and outbound types, where appropriate. In addition, types called **Partner** define a generic trading partner.

EDIFACT category

The **EDIFACT** category has four subtypes: **Control**, **Group**, **Msg**, and a **V** (*version*) category.

Control sub-tree

The subtypes of the **Control** category are the same in each **EDIFACT** tree. These types are used in **EDIFACT** envelope segments. Expand the **Control** category to view the subtypes.

Envelope types

Envelope types are used:

- To specify the syntax objects in the interchange that follows (the **UNA**).
- To surround messages or functional groups of an interchange (the **UNB** and **UNZ**).
- To surround a set of messages of the same type (the **UNG** and **UNE**).
- To surround the body of a message (the **UNH** and **UNT**).

Each **Envelope** type begins with a set of letters that identifies the segment, for example **UNA**. These same sets of letters (**UNA**) are also used in the Properties window of the Type Designer to define the value of the initiator.

UNA service string

The **UNA** type is an optional envelope control. It is used to specify syntax objects that appear within the EDIFACT interchange. The following are the components of the **UNA** type:

Composite Delimiter
A delimiter between components of a composite.

Element Delimiter
A delimiter between elements in an envelope or segment.

Decimal Delimiter
A separator between the integer and fractional part of EDIFACT decimal numbers.

ReleaseCharacter
A character used a release character.

Reserved Delimiter
Reserved for future use.

Terminator Delimiter
A terminator of an envelope or segment.

The **Composite**, **Element**, **ReleaseCharacter** and **Terminator Delimiter** types are all specified as syntax items. If a UNA appears in the data, the values that occur in the UNA set the active syntax values. If there is no UNA, the release character is assumed to be a question mark **?** and the delimiter values are set to the values in the UNB.

EDIFACT interchange envelope UNB and UNZ

An EDIFACT interchange begins with a UNB. It contains version release information, syntax information, and partner information. An interchange ends with a UNZ.

UNB is defined as a prefix delimited type with the initiator UNB. Its delimiter is defined as **Element Delimiter**. The first component of the **UNB** type specifies the character set for text items, the **Composite** delimiter and the version release.

Note: The default for the **UNB** delimiter is + and its default terminator is an apostrophe (') followed by a CR/LF. The default for the composite delimiter is set in the **SyntaxID Composite** type to a colon (**:**). You may change these defaults in the type tree, if necessary.

Functional group envelope UNG and UNE

A functional group begins with a **UNG** and ends with a **UNE**. Functional group envelopes in EDIFACT are optional. When they are used, the **Functional Group Ref# Elements** on both the **UNG** and the **UNE** must match. It is common not to use them, because the version information is also in the message envelope.

Message envelopes - UNH and UNT

A message begins with a **UNH** and ends with a **UNT**.

Element category

The item subtypes of the **Element** category are used as components of **Envelope** types. For example, trading partner identification items are defined under the **Element** category. The following **edif93_2.mtt** type tree displays a portion of the item subtypes of **Element**.

Delimiter category

Delimiter types are syntax items. They are used as composite delimiters, decimal separators, element delimiters, release characters, and terminators in the EDIFACT segments.

The values of these syntax objects may be different in different interchanges. Their values are determined in the **UNA**; each syntax object is a component of the **UNA** type. However, if the **UNA** is not present, the syntax values are found in the **UNB**. For example, the **UNB** type is defined as delimited, and the value of its delimiter is specified as the syntax item **Element Delimiter**. The value of **Find** property is **Yes**, indicating that, in input data, the value in the **UNB** is found and used as the value for the delimiter for the rest of the segments in the interchange.

Double-click a subtype of **Delimiter (for example, Element)** to view its restrictions. The restrictions are the values you can use for that syntax object. The following item window lists values to be included (considered to be valid) in a restriction list; they are the possible values of the delimiter between elements:

Composite category

A composite is a group of related elements. The **Composite group** subtypes are used as components of **Envelope** types. A composite has its own delimiter, which is typically a colon **:**. The following **edif93_2.mtt** type tree displays the subtypes of **Composite**:

Group sub-tree

The subtypes of the **Group** category represent functional groups in the given EDIFACT version.

The type name of a functional group is **F** followed by the EDIFACT version. For example, the functional group type in the **edif93_2.mtt** tree is **F93_2**.

A functional group is made up of messages. A functional group begins with a **UNG** and ends with a **UNE**, which are optional.

Msg sub-tree

The subtypes of the **Msg** category represent EDIFACT messages.

The subtypes of **F***version* (for example, **F93_2**) are all the messages in the given EDIFACT version. The name of each message is the value of the **MsgType Element** in the **UNH**.

A message begins with a **UNH** and ends with a **UNT**.

The middle component is the type that represents the actual contents of the message. These types are found under the version specific category, for example, the **V93_2** category in the **edif93_2.mtt** type tree.

Each message has a unique value for the **MsgType Element**. The value is specified by the component rule on the **UNH** component. For example, the component rule states that the **MsgType Element** of the **UNH** in the **BAPLTE** message is **BAPLTE**. Each message has a similar component rule.

Version specific sub-tree

The version specific category (for example, **V93_2**) is a subtype of the EDIFACT category. The name of this category is determined by the EDIFACT version.

Composite category

A composite is a set of related data elements. Each composite is defined as infix delimited and its delimiter is the syntax item **Composite Delimiter**. For a given interchange, the value of the **Composite Delimiter** is set in the UNA or the UNB. Typically, the composite delimiter is a colon.

Element category

The item subtypes of the **Element** category are the EDIFACT data elements. See the IBM WebSphere Transformation Extender Packs for EDI documentation for a

table listing the reference number, name of the element, and the abbreviation used in the EDI type trees. To see the full description of an element, view the type properties in the Type Designer.

To view an element's full description :

1. Select the type in the type tree.
2. Right-click and select **Properties** from the context menu. Or, click

   

   (**Properties**) on the tool bar.

   The description is located in the **Value** column of the **Description** property.

There are many element types in an EDIFACT tree. Although they are arranged alphabetically, you might want to use the Find command on the **Edit** menu to locate a particular element.

The following list summarizes how an EDIFACT type tree corresponds to the EDIFACT documentation of the EDI standards:

- Element codes are specified as restrictions in an EDIFACT tree.
- Different elements with the same EDIFACT description are named by appending the reference number designator to the end of the subtype name.
- Some elements have subtypes of their own. These will not be found as data elements in the EDIFACT documentation. **Element** subtypes were added if the same element was used more than once as a component of the same **Segment**. For example, if a **Segment** has two **Date Element** components, one might be an **Arrival Date Element** and the other a **Departure Date Element**.

Segment category

A segment is a logical unit of information, like a data record. The **Segment** type is partitioned by initiator value, each segment begins with a unique initiator, which identifies the segment. For example, the **BUS** segment begins with the initiator **BUS**.

Because an **Element Delimiter** must follow the segment initiator value and precede the first data element, each segment is defined as prefix delimited by the **Element Delimiter**, typically the plus sign + and has a terminator of **Terminator Delimiter**, typically the apostrophe (’). In addition, the release character is defined as **Release Character Delimiter**, typically the question mark (**?**).

Components of a segment may be **Element** types and **Composite** types. The segments were defined to comply with the rules specified in the EDIFACT documentation.

Note: For readability, the default **Segment** terminator has been defined as an apostrophe (’)**,** followed by a CR/LF. You can change this as needed.

Set category

The subtypes of the **Set** category define the contents of each message and the sets of repeating segment groups for the given EDIFACT version. Types are organized as subtypes of the **Partner** category: **Inbound** and **Outbound**. The subtypes of the **Inbound** and **Outbound** categories are identical.

Types defining message contents are organized under categories. Each type defining the contents of a certain message appears under a certain category. The name of the category is the message name, which matches the type name under the **Msg** category. For example, the type defining the contents of the **BAPLTE** message is found under the **BAPLTE** category.

The name of the type defining the contents of a message is **Message**.

Groups of segments

In EDIFACT, a repeating group of segments is called a group. A **BAPLTE** group type can also contain other group types. The word **Group** is typically appended with a number, such as 1 (for **Group1**), to distinguish it from other groups. For example, the **BAPLTE** message type contains **Group1**, **Group2,** and **Group3**.

Interchange category

Double-click the **Partner Inbound Interchange** type to view the components of the interchange.

The first component of the interchange, **UNA**, is sometimes called a ″service string″ in EDIFACT. It is optional. It specifies the syntax objects, such as delimiters, that appear in the rest of the interchange.

The second and fourth components collectively define the interchange envelope. The **UNB** is the envelope header. It specifies the trading partner information and syntactical information when the **UNA** is missing. The **UNZ** is the envelope trailer.

The third component defines the functional groups within the interchange. In EDIFACT data, the functional group envelopes are not required. Most often, EDIFACT data contains only a message envelope, which is inside the **Inbound Partner Group**.

The layout of components in the outbound interchange is similar to the layout in the inbound.

Restart attribute on functional group

The **Group** component of an interchange has the restart attribute. When you receive different functional groups from different departments within a trading partner, one department's good data is not lost if another department's data is bad. You can remove this restart attribute if you would rather go directly to the next interchange if bad data occurs within any **Inbound Group**.

Component rule on UNZ segment

The **UNZ** has a component rule that assures that the group of one interchange is not mixed up with the group of another interchange if a restart occurs.

The following is the component rule:

```
InterchangeControlRef Element:$ = InterchangeControlRef Element:Partner Inbound UNB
Envelope Control EDIFACT Data
```

The rule states that the **InterchangeControlRef Element** of the **UNZ** must be equal to the **InterchangeControlRef Element** of the **UNB**. If an **Inbound Partner Group** is invalid and a restart occurs, the component rule keeps different interchanges from getting mixed up with one another.

Transmission category

A **Transmission** type is made up of **Interchange** types. An **Inbound Transmission** type is made up of **Inbound Interchange** types. For example, an **EDIFACT Inbound Transmission** type is made up of **EDIFACT Inbound Interchange** types and a **Partner EDIFACT Inbound Transmission** type is made up of **Partner EDIFACT Inbound Interchange** types.

The **Partner** type represents a generic trading partner. If you are trading with different partners who use different subsets of the EDIFACT standards, you may want to rename the **Partner** type and add then add new types to define the additional trading partners.

The components of **Outbound Transmission** types are similar to those of **Inbound**.

The following group windows display the components of the **Inbound** and **Outbound** subtypes of **Transmission**. The **Partner Inbound Transmission EDI** group window illustrates that it is made up of **Partner Inbound Interchange** types and that the **Partner Outbound Transmission EDI** group window illustrates that it is made up of **Partner Outbound Interchange** types.

Restart attribute on an interchange

The **Interchange** component of a **Transmission** type has the restart attribute. On input data, the restart attribute is intended to filter network noise and to detect invalid interchanges. When you receive an invalid interchange from one partner, the restart attribute allows processing of valid data in other interchanges in the same transmission.

When you receive network messages and you know where they appear in an inbound communication file, you may want to include them as a component of an **Inbound Transmission** type. If you do not include them as data, they are not recognized as being part of a transmission and are rejected as invalid interchanges. However, the processing time is longer than if you included these network messages as data.

For EDI output data, the restart attribute is useful for auditing data. You can audit the output data to report when outbound interchanges within a transmission are invalid.

The following type tree, EdifISO9735-4.mtt conforms to ISO 9735 part 4.

Mapping EDIFACT data

When creating a map for EDIFACT data, create an input or output card to represent the EDIFACT data. When specifying the type of the card, select a subtype from the **Transmission** category.

When your input is EDIFACT data, select **Inbound Transmission**, or one of its subtypes. When sending EDIFACT data, select **Outbound Transmission**, or one of its subtypes.

Input card - EDIFACT data from multiple trading partners

When the input consists of only one interchange, select an **Interchange** type corresponding to the given **Transmission** type in the following table:

When You Receive	Select This Type
Multiple EDI standards	Inbound Transmission
Only EDIFACT data, and you have not defined different trading partners in your tree	Partner Inbound Transmission

See the IBM WebSphere Transformation Extender Packs for EDI for further information on creating EDI type trees.

When mapping to EDI data, select an **Outbound Transmission** type for the output card. The type to choose depends on what is being sent.

Output card - sending EDIFACT data to multiple partners

When the output consists of only one interchange, select an **Interchange** type corresponding to the given **Transmission** type in the following table:

If You Are Sending	Select This Type
Multiple EDI standards	Outbound Transmission
Only EDIFACT data, and you have not defined different trading partners in your tree	Partner Outbound Transmission

See the IBM WebSphere Transformation Extender Packs for EDI documentation for further information on creating EDI type trees.

Chapter 3. Example files

This chapter lists and describes the type trees and maps installed with the EDI Pack for EDIFACT.

EDIFACT example files

Example files for EDIFACT are located in the following folder:

*install_dir***packs****EDI_v***n.n****EDIFACT****examples****arrival**

The examples located in the **arrival** folder are reference tools that illustrate important concepts, as well as provide ideas that can be used in other maps.

The following map file and type trees install in the **arrival** folder:

- **arrival.mms**
- **simonedi.mtt**
- **simonin.mtt**

These files create an outbound EDIFACT Arrival Notice Message (IFTMAN) from multiple input files, a file of header records, and a file of detail records.

EDIFACT standard type trees

EDIFACT standard type trees install in the following folder:

*install_dir***packs****EDI_v***n.n****EDIFACT****trees**

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, a nd represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

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

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

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

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

Trademarks and service marks

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

AIX
AIX 5L
AS/400
Ascential
Ascential DataStage
Ascential Enterprise Integration Suite
Ascential QualityStage
Ascential RTI
Ascential Software
Ascential
CICS
DataStage
DB2
DB2 Universal Database
developerWorks
Footprint
Hiperspace
IBM
the IBM logo
ibm.com
IMS
Informix
Lotus
Lotus Notes
MQSeries
MVS
OS/390
OS/400
Passport Advantage
Redbooks
RISC System/6000
Roma
S/390
System z
Trading Partner
Tivoli

WebSphere
z/Architecture
z/OS
zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

IBM WebSphere Transformation Extender Pack for EDIFACT, Version 2.7

Index

A

arrival folder 11
arrival.mms 11

B

BAPLTE
  group type 8

C

categories
  Composite 6
  Delimiter 5
  Element 5, 6
  Group 6
  Interchange 8
  Set 7
  Transmission 9
  version specific 6
category types
  Control 4
  EDIFACT 3, 4
  Interchange 3
  Transmission 3
component rule 8
Composite
  category 6
  definition of 6
Control
  category type 4

D

Delimiter
  category 5
  restrictions 5
description
  element 6

E

EDIFACT
  category type 3, 4
  examples 11
Element
  category 5, 6
Envelopes
  group type 4
  UNA 4
  UNB 4
  UNE 4
  UNG 4
  UNH 4
  UNT 4
  UNZ 4
examples
  directory 11

F

functional groups
  envelopes 5

G

Group
  category 6
group types
  Envelope 4

I

input card 10
  settings 9
installation 1
Interchange
  category 8
  category type 3
  restart attribute 9
  UNA 8
Introduction 1

M

map file 11
mapping 9
message envelopes 5

O

output card 10

R

restart attribute 9
  functional group 8

S

segments
  definition of 7
  groups 8
Set
  category 7
simonedi.mtt 11
simonin.mtt 11

T

Transmission
  category 9
  category type 3
  restart attribute 9
type trees 11

U

UNA 4, 8
components of 4
UNB 4, 5
UNE 4, 5, 6
UNG 4, 5
UNH 4, 5, 6
UNT 4, 5, 6
UNZ 4, 5, 8

V

version specific
category 6

Printed in USA