

Version 3.3



Mapping Guide

Version 3.3



Mapping Guide

Note!

Before using this information and the product it supports, read the information in “Notices” on page 345.

March 2007

This edition applies to IBM WebSphere Data Interchange for MultiPlatforms, V3.3. and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this documentation, email doc-comments@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.

© Copyright International Business Machines Corporation 2007. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	xi
Tables	xiii
About this book	xv
Who should read this book	xv
How this book is organized	xv
Related books	xv

Part 1. Introduction 1

Chapter 1. Introducing WebSphere Data Interchange Mapping	3
Prerequisites	3

Chapter 2. Understanding Mapping	5
Mapping process overview	6
How mapping works	6
Map types	7
Data Transformation maps	9
Validation maps	10
Functional acknowledgement maps	10
Send maps	11
Receive maps	11
Object types	12
Code lists	12
Translation tables	12
Global variables	13
Control strings	14
Queries	15

Chapter 3. The WebSphere Data Interchange mapping interface	17
Main application window	17
Mapping functional area	18
XML functional area	20
EDI functional area	21

Chapter 4. Data formats	25
Creating a data format	26
Understanding how your application data is structured	26
Filling out a data format worksheet	31
Using the WebSphere Data Interchange data format editors	33
Accessing data format editors	34
Creating a data format dictionary	35
Creating the data format record ID information	38
Creating a data format	40
Creating loops	43

Creating a data format record	45
Creating a data format structure	47
Creating a data format field.	49
Navigating between data format editors	50
Data format dictionary editor paths	50
Data format editor paths.	51
Data format loop editor paths	52
Data format record editor paths	52
Data format structure editor paths	53
Data format field editor paths	53
Reusing data format components.	53
Understanding data types	54
Chapter 5. Extensible Markup Language	61
Accessing XML editor	62
Creating the XML dictionary	62
Importing and defining a DTD or schema file	63
Creating an XML Namespace	65
XML Document processing.	66
Example 1	67
Example 2	68
Example 3	69
Chapter 6. EDI standards	71
Envelopes	71
Transactions	72
Using the EDI standard editors	73
Creating the EDI standard dictionary	74
Creating a transaction	76
Creating a segment	79
Creating a data element.	81
Creating a code list	84
Editing an envelope standard	84
Envelope control strings.	87
Chapter 7. Creating map objects	89
Creating code lists	89
Creating translation tables	90
Creating global variables	90
Chapter 8. Creating a map	93
Using the map editor	93
Starting the map editor	95
General editing procedures.	95
Advanced mapping techniques	100
Mapping task list.	100
Choosing the right map	102
Specifying qualification	103
Qualifying a Data Transformation map.	104
Qualifying a send or Receive map	110

Qualifying data elements	114
Compiling control strings	116
Recompiling control strings	116
Migrating a map	117

Part 2. Data Transformation Maps 119

Chapter 9. Data Transformation mapping	121
Using the Data Transformation Map editor	121
Creating a new Data Transformation map	122
Mapping hierarchical loops	123
XML mapping considerations	123
Namespaces	124
Target Namespace	124
Namespace Processing for Input XML documents	124
Namespace Processing for Output XML Documents	125
XML schema restrictions	125
Applying map rules	125
Applying the minimal trading partners concept	125
Viewing map rules	126
Mapping MQMD and MQRFH2 values	126
Getting and setting properties in the MQMD and MQRFH2 headers	126
Other notes	129
Chapter 10. Validation mapping	131
Starting the validation map editor	131
Creating a validation map	132
Using validation maps	132
Chapter 11. Functional Acknowledgement mapping	135
Functional Acknowledgement maps provided with WebSphere Data Interchange	135
Starting the Functional Acknowledgement map editor	136
Using the Functional Acknowledgement map editor	137
Creating a Functional Acknowledgement map	137
Source document definition record layout	138
Using Functional Acknowledgement Maps	140
Chapter 12. Data Transformation mapping commands and functions	141
Map variables	141
Naming variables	142
Literals	142
Comments	143
Keywords	143
Specifying a path	143
Forward and reverse references	144
Data types supported by mapping commands and functions	144
Expressions	145
Logical operators	145
Comparison operators	146

Arithmetic operators	146
Unary operators	147
Order of precedence	147
Assignment	147
Conditional commands	147
If / Elself / Else / Endif	147
Commands	149
CloseOccurrence	149
Create	150
Error	150
ErrorContext	151
FAError	151
FAErrorPath	155
ForEach	155
HLLevel	158
MapCall	158
MapChain	159
MapFrom	159
MapSwitch	160
MapTo	161
Qualify and Default	162
SetElementAttribute	163
SetNamespace	172
SetNoNSSchemaLocation	173
SetSchemaLocation	173
SetProperty	174
Functions	175
Char	175
Concat	176
Created	176
Date	177
DateCnv	177
Exit	177
Find	178
Found	178
GetProperty	179
HexEncode	179
HexDecode	180
IsEmpty	180
Left	181
Length	181
Lower	182
Number	182
NumFormat	182
Occurrence	183
Overlay	184
Right	185
Round	185
StrComp	186
StrCompl	186

StrCompN	187
StrCompNI	187
SubString	188
Time	189
Translate	189
TrimLeft	190
TrimRight	191
Truncate	191
Upper	192
Validate	192
Message properties	193
Source document properties	193
Target document properties	194
EDI envelope standard generic properties	195
EDI envelope standard specific properties	196

Part 3. Send and Receive Maps 199

Chapter 13. Send and Receive mapping	201
Creating a send or Receive map	201
Using the mapping data element editor	202
Applying advanced mapping capabilities to a field or data element	203
Using the special handling button	203
Using the Repeat Button	209
Setting an application control key	209
Using literals and mapping commands	211
Adding a literal or mapping command to a map	211
Literals and data types	211
Translation tables	212
Forward translation tables	213
Reverse translation tables	213
Creating a new translation table	214
Specifying send and receive usages	215
Applying the minimal trading partners concept	215
Viewing usages	216
Creating a Send map usage	216
Creating a receive usage	225
Editing send and receive usages	238
Copying send or receive usages	238
Defining generic send usages	238
Defining generic receive usages	239
Creating fixed-to-fixed maps	240
Chapter 14. Advanced send and Receive mapping	241
Using accumulators	241
Accumulator types	242
Accumulator actions	242
Adding an accumulator to a map	242
Using literals	243

Using literals for Send mapping	243
Segment creation for Send mapping	244
Using literals for Receive mapping	244
Format of literal data	245
Accumulator literals	245
Conditional processing of literals	245
Literal keywords	246
Named variables	256
Expressions	258
Boolean operators	259
Comparison operators	259
Arithmetic operators	259
Unary operator	261
Special operators	261
Date conversion special operators	263
Order of precedence	265
Special variables	266
Mapping techniques for literal keywords	268
Examples of using literal keywords and named variables	270
Example 1	270
Example 2	270
Example 3	270
Example 4	271
Example 5a	271
Example 5b	272
Example 6	272
Notes on examples 5 and 6	273
Example 7	273
Example 8	274
Example 9	275
Example 10	276
Control data literals	277
Using service segment fields	277
Mapping service segment fields (send only)	278
Mapping specific service segment fields (receive only)	279
Mapping generic service segment fields (receive only)	280
Validation during mapping	282
Appendix A. Mapping Binary Data	285
The BIN segment ID	285
Length of the BIN segment	285
Data Transformation for Binary Data	285
Mapping a BIN segment	286
The BIN and BDS segments	286
The EFI segment	287
Send processing for the binary segment	288
Mapping data from a file to a binary segment	288
Format specifications	289
Examples	290
Mapping an application field to a binary segment	291

Receive processing for the binary segment	293
Mapping data from a binary segment to a file	293
Mapping data from a binary segment to an application field	296
Appendix B. Hierarchical loops	297
Specifying HL levels	297
The HL segment	298
Preparing hierarchical loops	299
Data Transformation Mapping for HL Loops	300
Creating a Data Transformation Map	300
Defining HL loop levels.	300
Qualifying HL loop levels in an EDI source message	301
Qualifying HL loop levels in an EDI target message	301
Handling special HL mapping for Data Transformation maps	302
Mapping the HL segment in a send or Receive map	310
HL segment literal keywords for Send and Receive Maps	311
Appendix C. Handling international characters	313
Processing and automatic detection for UNICODE	314
Encoding used for incoming Application data	314
Encoding used for outgoing Application data	314
Encoding used for incoming EDI data	314
Encoding used for outgoing EDI data	315
Encoding used for incoming XML data	315
Encoding used for outgoing XML data.	315
Import and export considerations	316
MQ profile fields	316
PERFORM keywords for international data	316
SOURCEENCODE and ENCODETARGET	316
IGNOREBOM.	317
Byte-order mark support	317
Examples	319
Scenario 1: WebSphere Data Interchange in a worldwide data center	319
Scenario 2: Exporting data from one database and importing it into another	319
Scenario 3: Migrating from a previous version of WebSphere Data Interchange	320
Appendix D. DTD Conversion Utility	321
Converting a DTD	321
DTD Conversion Utility output	325
XML Dictionary File	326
Resolving the XML dictionary file name	326
Updating a previously converted DTD	327
Processing external DTDs.	327
Resolving DTD file names.	328
Example of DTD file resolution	328
Using a DTD alias file	329
Inbound Translation Process (XML to Data Format)	329
Receiving XML data using VAN (inbound)	330
Outbound Translation Process (Data Format to XML).	331
Sending XML data using VAN (outbound)	332

Specifying Sender and Receiver Information	332
Identifying trading partners	334
PERFORM Commands and Keywords	334
Encoding considerations	336
Overriding the Default XML Prolog	337
Mapping considerations for XML data	337
EDI Standard Representation of XML Data	338
Sequences.	339
Choices.	339
Special Sequences and Choices	340
Mapping example	340
Control String Generation	341
Diagnosing Errors	341
Understanding DTD parsing	341
Parser messages	341
Warning messages	342
Utility JCL	343
XML Processor Messages and Codes.	344
Notices.	345
Programming interface information	347
Trademarks and service marks	347
Glossary of terms and abbreviations	349
Bibliography	357
WebSphere Data Interchange publications	357
Softcopy books	357
Portable Document Format (PDF)	357
WebSphere Data Interchange information available on the Internet	357
Index	359

Figures

1. Mapping process flow	6
2. Mapping functional area	20
3. XML functional area	21
4. EDI functional area	22
5. Data formats functional area	26
6. Application data with record identifier	27
7. Sample Working Storage Record Definition of a Purchase Order	31
8. Data Transformation Map editor, Details tab	96
9. Example of a Data Transformation Map	97
10. Cascading menus in the Map Command window pane	99
11. Data Element Special Handling window	204
12. Send Map Usage editor	216
13. Receive Map Usage editor	225
14. HL example 1	297
15. HL example 2	297
16. The HL segment	298
17. Preparing Hierarchical Loops	300
18. Example of EDI no hierarchy (no parent id HL02)	302
19. XML DTD with nesting	303
20. XML DTD without nesting	304
21. Loop menu	305
22. HL Qualification window	305
23. HLLevel command	306
24. A child HL qualification	307
25. A peer (sibling) HL qualification	307
26. No qualification under HLLEVEL data example	308
27. CloseOccurrence command with no qualification example	309
28. CloseOccurrence command with no qualification incorrect data example	309
29. Multi-occurrence qualification example	310
30. Multi-occurrence qualification data example	310
31. DTD Conversion Utility	321

Tables

1. Map types supported by WebSphere Data Interchange	8
2. Object types	12
3. Mapping functional area components	19
4. Sample Application Records with C and D Records	28
5. Sample Record	30
6. Data Format Component Relationship Worksheet Example	31
7. Data Format Component Relationship Worksheet	33
8. Data format symbols	51
9. Data types for data formats	54
10. Add EDI Standard Transaction Detail editor fields	77
11. Add EDI Standard Transaction Segment editor fields	80
12. Add EDI Standard Transaction Segment editor fields	82
13. Data types for EDI standard data elements	83
14. Add EDI Standard Transaction Segment editor fields	85
15. Symbols used in maps	98
16. Control string list window field descriptions	116
17. MQMD properties (ROOT.MQMD.xxx)	127
18. MQMD properties for Windows and AIX	128
19. MQRFH2 properties (ROOT.MQRFH2.xxx)	128
20. Standard Functional Acknowledgement maps	135
21. Functional acknowledgement dictionaries	136
22. Supported special variables	142
23. Comparison operators	146
24. Format examples for SuppressZeroValues	171
25. Format examples for SuppressZeroValues	172
26. Special Handling Options	204
27. Translation table, differences in data	212
28. Translation table, conflicts with standards	212
29. Sample forward translation, table values	213
30. Sample reverse translation, table values	213
31. Send map usages Exit Routines tab fields descriptions	219
32. Send map usages Envelope Attributes tab field descriptions	220
33. Send map usages WDI options tab field descriptions	222
34. Application Routing field descriptions	228
35. Receive map usages Attributes tab field descriptions	232
36. Receive map usages WDI Options tab field descriptions	234
37. WebSphere Data Interchange accumulator types	242
38. Accumulator actions	242
39. Literal keywords	246
40. Sample translation table UOMDIV	275
41. Literals to identify fields in service segments	279
42. Keywords for mapping envelope data (receive only)	280
43. Validation used during mapping for different data types	282
44. Type of validation during Translate to Standard	282
45. Type of validation during Translate to Application	283
46. The HL segment	299
47. HL segment literal keywords	311

48. WebSphere Data Interchange supported BOMs.	318
49. DTD Conversion Utility field descriptions	322

About this book

This document explains how to use WebSphere® Data Interchange Client V3.3 to develop Data Transformation, validation, functional acknowledgement, and send and Receive maps to use when processing EDI documents, XML documents, and data formats.

Who should read this book

This document is intended for the person responsible for creating Data Transformation maps, validation maps, functional acknowledgement, and send and Receive maps using WebSphere Data Interchange.

How this book is organized

This book has the following three parts:

- Part 1. Introduction
This part contains information about the WebSphere Data Interchange mapping functional area. It includes information about how to create data formats, EDI standards, XML dictionaries, map objects.
- Part 2. Data Transformation maps
This part covers the information you need to create Data Transformation, validation and Functional Acknowledgement maps.
- Part 3. Send and Receive maps
This part covers the information you need to create send and Receive maps.

Related books

The following books complete the WebSphere Data Interchange library and contain information related to the topics covered in this book. You can view these documents, and download them, from the library page of the WebSphere Data Interchange Web site:

<http://www.ibm.com/websphere/datainterchange>

- *WebSphere Data Interchange for MultiPlatforms Quick Start Guide*, CF0YREN
This document provides a brief overview of how to use WebSphere Data Interchange.
- *WebSphere Data Interchange for MultiPlatforms Administration and Security Guide*, SC34-6214-01
This document provides information on administrative tasks you will use in WebSphere Data Interchange.
- *WebSphere Data Interchange for MultiPlatforms Messages and Codes Guide*, SC34-6216-01
This book provides information to assist you in diagnosing errors.
- *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01

About this book

This book provides information on the WebSphere Data Interchange Client/Server user interface.

- *WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide*, SC34-6217-01

This document provides detailed technical information about WebSphere Data Interchange.

- *WebSphere Data Interchange for MultiPlatforms Mapping Guide*, SC23-5874-00

This document provides instructions for your WebSphere Data Interchange mapper.

- *WebSphere Data Interchange for MultiPlatforms Utility Commands and File Formats Reference Guide*, SC23-5873-00

This document provides the commands and file formats necessary to use WebSphere Data Interchange.

- *WebSphere Data Interchange for z/OS V3.3 Installation Guide*, SC34-6269-01

This book provides information for the electronic data interchange (EDI) administrator about entering, sending, and receiving EDI transactions and other documents interactively.

Part 1. Introduction

Chapter 1. Introducing WebSphere Data Interchange Mapping

Before you can use WebSphere Data Interchange to translate data, or to send or receive transactions, messages, or files, you must define certain information. This information describes how your system sends and receives data, how data is formatted in your application files and to a standard, to whom you send data and from whom you receive data, and other pertinent information.

This document explains how to use WebSphere Data Interchange to develop Data Transformation, validation, functional acknowledgement, and send and Receive maps to use when processing Electronic data interchange (EDI) documents, Extensible Markup Language (XML) documents, or proprietary documents (data format).

This document outlines the following items required for mapping:

- Proprietary documents (data formats)
- XML
- EDI Standards

You will also find information about creating maps and specific instructions for creating:

- Data Transformation maps
- Send maps
- Receive maps

Prerequisites

This document is intended for the person responsible for creating Data Transformation maps, validation maps, Functional Acknowledgement maps, send and Receive maps using the WebSphere Data Interchange Client.

Before using this document, familiarize yourself with the WebSphere Data Interchange Client and concepts of use in the *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Chapter 2. Understanding Mapping

A WebSphere Data Interchange map relates a source document to a target document. In WebSphere Data Interchange, there are three types of documents:

- EDI

An EDI document can be a transaction such as a purchase order or an invoice in a format defined by a standards body such as X12 or EDIFACT.

- XML

An XML document can be any type of information encoded using Extensible Markup Language tags according to an XML schema or DTD.

- Proprietary documents (data formats)

A proprietary document (data format) can be one of many types of formats used by data processing applications.

Mapping process overview

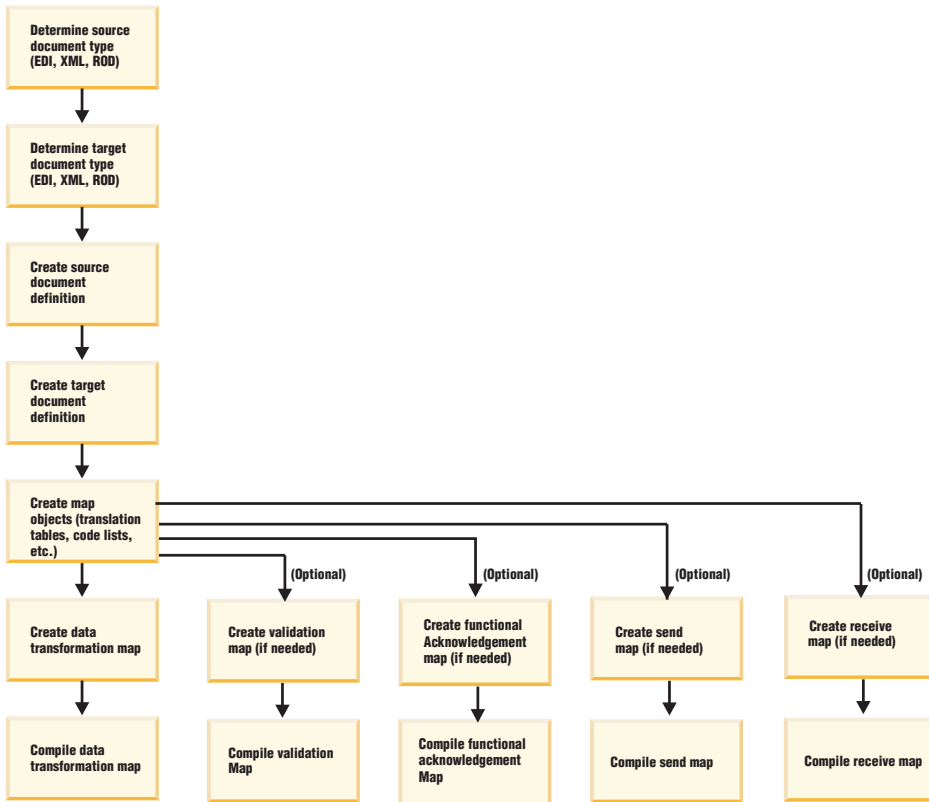


Figure 1. Mapping process flow

How mapping works

In WebSphere Data Interchange you can create or import document definitions for the source and target documents, and then create a map which relates the elements in the source document to elements in the target document.

Any combination of source and target document types can be specified in a map. For example, you can create a map which transforms an EDI document into an XML document, or proprietary documents (data formats) into an EDI document. Similarly, EDI-to-EDI and XML-to-XML maps are possible, along with the other combinations.

A map contains the commands necessary to transform a source document into a target document. WebSphere Data Interchange provides a graphic user interface (GUI) which simplifies creation of the mapping commands. In the simplest cases, you can drag-and-drop mapping between source and target elements. This type of mapping handles cases where there is a one-to-one correspondence between source and target

elements. You can also drag elements when creating more complex mapping commands. After completing the map, you compile it into a control string using WebSphere Data Interchange Client. In addition to the Data Transformation maps described, there are four types of specialized maps:

- Validation maps

A validation map provides the instructions needed to perform additional validation beyond what is specified in the EDI Standard. In a validation map, there is no target document. A special command (**FAError()**) is provided to report any errors in an EDI functional acknowledgement. You can create your own validation maps. Special *service segment* validation maps are provided with the product. The service segment validation maps validate the envelope segments such as the X12 ISA or the EDIFACT UNB. (These are called control segments in X12 and service segments in EDIFACT.) Both service segment validation and a user-specified validation map can be applied to a given EDI document.

- Functional acknowledgement maps

A Functional Acknowledgement map is a special Data Transformation map used in creating EDI acknowledgements. EDI acknowledgements include the X12 TA1 interchange acknowledgement and the X12 997 and EDIFACT CONTRL functional acknowledgements. WebSphere Data Interchange creates all EDI acknowledgements using a single data format definition. A Functional Acknowledgement map is used to convert this document into an EDI document such as an X12 997 transaction set or an EDIFACT CONTRL message. Several standard Functional Acknowledgement maps are provided with the product.

- Send maps

A Send map specifies how a field in a Data Format is used to create a Data Element in a Transaction. In Send Maps, the source document is always a data format. The target document is always an EDI Standard Transaction. Send Maps are target based. This means that commands are executed in an order based on the target document definition (the EDI Standard Transaction).

- Receive maps

A Receive map specifies how a field in a Data Format is created from a Data Element in a Transaction. In Receive Maps, the target document is always a data format. The source document is always an EDI Standard Transaction. Receive Maps are source based. This means that commands are executed in an order based on the source document definition (the EDI Standard Transaction).

Map types

WebSphere Data Interchange supports five types of maps that you can use to transform data that you exchange with other business and trading partners: Data Transformation maps, validation maps, Functional Acknowledgement maps, and send and Receive maps.

The following table describes each type of map supported by WebSphere Data Interchange.

Understanding mapping

Table 1. Map types supported by WebSphere Data Interchange

Map type	Description
Data Transformation	Data Transformation maps are the primary type of map used to convert a document from one format to another. They provide the information needed to gather data from a source document and create a target document using the source data. These maps provide a powerful method of mapping data from one document format to another.
Validation	Validation maps provide the instructions needed to perform additional validation beyond what is specified in the EDI Standard. Validation maps contain mapping commands that are instructions used to provide additional validation of an EDI document.
Functional acknowledgement	Functional acknowledgement maps provide the instructions to the WebSphere Data Interchange on how to produce a functional acknowledgement. A Functional Acknowledgement map is a special Data Transformation map used in creating EDI acknowledgements.
Send	Send maps convert a Data Format source document to an EDI Standard Transaction target document.
Receive	Receive maps convert an EDI Standard Transaction source document to a Data Format target document.

All five map types provide the ability to perform the following tasks:

- Directly relate two elements (Data Transformation maps only)
- Generate custom functional acknowledgements (Functional Acknowledgement maps only)
- Generate extended validation (validation maps only)
- Manipulate data
- Set fixed values into elements
- Validate data
- Change values of data based on equivalent value specified in a translation table
- Specify conditional translation processing
- Save data or set flags for use in other translations
- Handle repeating compound or simple elements in similar or different ways (send and Receive maps do not support repeating elements in EDI)
- Handle hierarchical loops (HL's)

Data Transformation, functional acknowledgement, and validation maps provide the ability to perform the following tasks:

- Directly relate two elements (Data Transformation maps only)
- Generate custom functional acknowledgements (Functional Acknowledgement maps only)
- Generate extended validation (validation maps only)
- Provide comments anywhere in the map
- Group commands and comments together
- Copy or move mapping commands and comments

- Switch to a different map or execute another map after the current one is finished

Send and Receive maps provide the ability to perform the following tasks:

- Comments at the mapping level only
- Move mapping commands within an element (no copy is available for send and Receive maps)
- For Receive maps only, switch to a different map or execute another map after the current one is finished

Data Transformation maps

Data Transformation maps consist of commands specifying how to transform a source document into a target document. In each Data Transformation map, a source document definition and a target document definition is specified. The source document definition describes the layout of the source document (the document to be translated). The target document definition describes the layout of the target document (the output document). Source and target document definitions are specified when the map is first created. You can change the source and target document definitions after a map is created, but you cannot change the source syntax type or target syntax type. For instance, if you are using an EDI document definition as your source document definition, you can change which version of the EDI Standard you are using, but you cannot indicate that you want the source document syntax type to be a data format instead of EDI.

You can indicate whether a Data Transformation map is source or target-based when you create it. This cannot be changed after the map is initially created.

- Source-based

A Data Transformation map can be source-based. In a source-based Data Transformation map, the map is constructed based on the order elements are defined in the source document definition. This provides an efficient and predictable method of processing the data and the mapping commands. In a source-based map, the source document definition is the base document definition. If the source document contains hierarchical loops and you need to use the special hierarchical loop mapping command (`HLLLevel()`) in a Data Transformation map to construct the target document, the map must be source-based.

- Target-based

There are cases where it is easier to produce the target document if the mapping commands are executed based on the order elements are defined in the target document definition. Data Transformation maps support target-based processing. If the target document contains hierarchical loops and you need to use the special hierarchical loop mapping command (`HLLLevel()`) in a Data Transformation map to construct the target document, the map must be target-based. Generally, source-based maps are used whenever convenient because translation based on the source document is usually more efficient than translation based on the target document. In a target-based map, the target document definition is called the base document definition.

Understanding mapping

Validation maps

Validation maps provide the instructions needed by WebSphere Data Interchange to perform additional validation beyond what is specified in the EDI Standard.

A validation map is similar to a source-based Data Transformation map. Unlike a Data Transformation map, however, a validation map has no target. Certain specific commands such as **MapTo()** are not available in a validation map. The **FAError()** command is available in a validation map for reporting errors in an EDI acknowledgement.

Each validation map has an associated source document definition. The source document definition describes the layout of the document to be validated. It is specified when the map is first created. The source document definition is always an EDI document definition. You can change the source document definition after a map is created, but you cannot change the source syntax type. For instance, you can change which version of the EDI Standard you are using, but you cannot indicate that you want the source document type to be a data format instead of an EDI Standard.

Functional acknowledgement maps

Functional acknowledgement maps provide the instructions to the WebSphere Data Interchange on how to produce a functional acknowledgement.

WebSphere Data Interchange can automatically generate EDI Standard functional acknowledgements for EDI documents received from a trading partner. To produce a functional acknowledgement, a Functional Acknowledgement map must be in the Map Rule associated with the source document that is being translated. The source document must be an EDI Standard Transaction. When a document is going to be translated, it is first validated as specified in the Map Rule associated with the translation. WebSphere Data Interchange will provide a standard level of validation on the EDI Standard Transaction document. If a functional acknowledgement is going to be generated, results from validation of an EDI Standard Transaction are written to an internal format. Validation Maps are created to provide additional validation on an EDI Standard Transaction document. Results are also written to the internal format. The generation of a functional acknowledgement uses the Functional Acknowledgement Map specified in the Map Rule and this internal format as its source document. The Functional Acknowledgement Map contains mapping commands that indicate how to use the validation results contained in this internal format to create a specific functional acknowledgement. If a document is accepted for translation by the validation process, then the appropriate Data Transformation Map is used to translate the source document. All of this is controlled by the Map Rule associated with the source document and the Data Transformation Map.

The source document definition for all Functional Acknowledgement Maps is a Data Format with the name &FUNC_ACK_META contained in the dictionary &FUNC_ACK_METADATA_DICTIONARY. It describes the layout of the internal format generated by the validation process. The name of the source document definition cannot be changed. You cannot create a Functional Acknowledgement Map without this Data Format in your System.

The target document definition in a Functional Acknowledgement Map describes the layout of the functional acknowledgement. It must be an EDI Standard Transaction with a name of 997, 999 or CONTRL. Normally, you do not need to create or modify a Functional Acknowledgement map. WebSphere Data Interchange provides several maps to produce the most common functional acknowledgements. Create a Functional Acknowledgement map only when a custom functional acknowledgement is required. For a list of the Functional Acknowledgement maps provided by WebSphere Data Interchange, see Table 20 on page 135.

Send maps

Send maps provide instructions to WebSphere Data Interchange for translating a document from a proprietary format into an EDI Standard Transaction. Send maps also provide the information needed by WebSphere Data Interchange to gather data from a document in a proprietary format and create a Transaction using the source data. Send maps offer a powerful method of mapping data from a proprietary format to an EDI Standard Transaction.

In addition to having instructions that specify how to convert a document from one format to another, WebSphere Data Interchange must also know the layout, or format, of the source and target document. In WebSphere Data Interchange, the layout of a document is a document definition. Send Maps use an EDI Standard Transaction as the target document definition. The source document definition, the layout of the proprietary format, is called a Data Format. Maintenance of document definitions is not performed in maps or even in the Mapping Functional Area. Instead, each of the document syntax types is maintained in their respective Functional Areas. In maps, you are always working with document definitions; never on the documents themselves.

Send maps are used to show how a field in a Data Format is used to create a Data Element in a Transaction. In Send Maps, the source document is always a data format. The target document is always an EDI Standard Transaction. Send Maps are target based. This means that commands are executed in an order based on the target document definition (the EDI Standard Transaction). Through mapping, you specify the relationships between the fields in a Data Format and Data Elements in a Transaction.

Receive maps

Receive maps provide instructions to WebSphere Data Interchange for translating a document from an EDI Standard Transaction into a proprietary format. Receive maps also provide the information needed by WebSphere Data Interchange to gather data from a Transaction and create a document in a proprietary format using the source data. Receive maps offer a powerful method of mapping data from an EDI Standard Transaction to a proprietary format.

In addition to having instructions that explain how to convert a document from one format to another, WebSphere Data Interchange must also know the layout, or format, of the source and target document. In WebSphere Data Interchange, the layout of a document is a document definition. Receive Maps use an EDI Standard Transaction as the source document definition. The target document definition, the layout of the proprietary format, is called a Data Formats. Maintenance of document definitions is not performed in maps or even in the Mapping Functional Area. Instead, each of the

Understanding mapping

document syntax types is maintained in their respective Functional Areas. In maps, you are always working with document definitions; never on the documents themselves.

A Receive Map is used to show how a field in a Data Format is created from a Data Element in a Transaction. In Receive Maps, the target document is always a data format. The source document is always an EDI Standard Transaction. Receive Maps are source based. This means that commands are executed in an order based on the source document definition (the EDI Standard Transaction). Through mapping, you specify the relationships between the fields in a Data Format and Data Elements in a Transaction.

Object types

WebSphere Data Interchange provides the following types of objects that you can use when creating or editing maps: code lists, transformation tables, and global variables.

The following table describes each available object type.

Table 2. Object types

Object type	Description
Code list	A code list defines a list of acceptable values that are used in maps to validate any source simple element or variable.
Translation table	A translation table is used to change one value into another corresponding value during translation of a document.
Global Variable	Global variables are used with Data Transformation maps, validation maps, and Functional Acknowledgement maps. During translation, you can put data into a global variable. The data contained in a global variable is available to other translations.

Code lists

A code list is a list of acceptable values that is used in maps to validate any source simple element or variable. Code lists are provided by EDI Standards. They indicate which values are valid for a specific EDI data element. A code list can also be created to specify any list of valid values. When WebSphere Data Interchange validates or translates a document, and the use of a code list is specified to validate a piece of data, the specified code list is searched to see if it contains the value.

Translation tables

A translation table is used to translate a value into a corresponding value. Translation tables are used in Data Transformation maps, validation maps, and Functional Acknowledgement maps. Translation tables contain a list of unique values called source values. Each unique source value has a corresponding target value assigned to it. The target values do not need be unique. Always define all possible values in the translation table.

Use a translation table in a map to indicate that an alternate value is to be used instead of the value contained in a simple element or variable. An example of where a translation table might be useful is when a simple element contains an internal part number that must be converted to a trading partner's corresponding part number. When the instruction to use a translation table is encountered during translation, the value in the source simple element is looked up in the specified translation table. If it is found in the table, the corresponding value is then substituted for the original source value. If it is not found, there is an option to specify a default value.

In Data Transformation maps, validation maps, and Functional Acknowledgement maps, the lookup of a value is normally done against the source values in a translation table. When a matching source value is found, the corresponding target value is substituted for the original value. There is an option to perform a lookup using the target value. With this option, you can use the same table to translate a value to a trading partner's equivalent value and also to translate a trading partner's value to your equivalent value. Using the target value for a lookup is useful only when target values are unique. When the option to perform a lookup using the target value is used, the lookup attempts to locate a target value in the translation table that matches the value in the simple element or variable. When found, the corresponding source value is substituted for the original value.

Global variables

A global variable defines a variable that can be used across mapping translations. These variables are used much like variables in most programming languages. Global variables can hold and manipulate data in Data Transformation maps, validation maps, and Functional Acknowledgement maps. While a document is being translated, data can be put into a global variable. After the translation of the document ends, the data remains in the global variable. The data in the variable might be available in the next translation, depending on the scope of the variable, regardless of the map that is used to perform the translation. During subsequent translations within the scope of the variable, the data in the global variable can be obtained, manipulated and changed.

One attribute of a global variable is scope. The scope of a global variable can be session, group, or interchange. Global variables that have a scope of interchange are created and initialized when an interchange is encountered. They are reset at the start of the next interchange. Variables with a scope of group are created and initialized when a group is encountered. They are reset at the start of the next group. Not all syntax types support the concept of groups or interchanges. When the source document type does not support groups, the scope of group is treated the same as interchange. If the document type does not support interchanges, the variable is reset at the start of each input document. For example, if the input file contains multiple documents and is split into separate documents by the Splitter or XMLSplitter, the global variables are reset for each input or XML document in the file.

Other attributes of global variables include data type, maximum length, and initial value. Maximum length dictates how long the data within a global variable can be. The maximum length is only available for certain data types. Global variables can also have an initial value. When a global variable is created or reset automatically (goes out of

Understanding mapping

scope), the initial value will automatically be put into the global variable. The default initial value for a numeric data type is zero. An empty string is the default initial value for character string and binary data types.

Data Transformation maps, validation maps, and Functional Acknowledgement maps use global variables in map commands. Every global variable is available to every map. A global variable must be defined before it can be used in a map command. They can be defined from the Global Variables list window of the Mapping Functional Area, the Data Transformation Map editor, the Validation Map editor, or the Functional Acknowledgment Map editor.

A Global Variable must reside on the WebSphere Data Interchange Server before a map using the Global Variable can successfully translate a document. Ensure that the Global Variable exists on the server by defining the Global Variable on the server using WebSphere Data Interchange Client or use export functions to copy the Global Variable to the server. Global Variables can be exported by themselves or with maps or Map Control Strings.

Note: Certain changes to a global variable affect all maps that utilize the global variable. All maps that use a global variable that have had certain changes must be recompiled using WebSphere Data Interchange Client. A map must be recompiled when the name, scope, or data type of a global variable has been changed.

Control strings

Control strings help improve the performance of the WebSphere Data Interchange Server when translating a document. Control strings are the compiled representation of a map. On the server, the original maps are never used. Instead, a map must be compiled into a control string before the WebSphere Data Interchange Server can translate the source document referenced by the map. Map Control Strings are required for all map types.

When you compile a map using WebSphere Data Interchange Client, control strings are generated for the map and often for the source and target document definitions referenced by the map. The map control strings associated with compiled maps can be listed on the Control Strings list window in the Mapping Functional Area. Control strings associated with document definitions cannot be viewed. These are managed with the map control strings. Document definition control strings are exported with the map control strings that need them.

Map control strings are recompiled any time the corresponding map is changed or any time the related source or target document definition is changed. Changes will not be available to WebSphere Data Interchange until the recompile has been successfully completed.

Queries

A query is a request for specific information from the database. By using queries, you can determine what information you want to see and the order in which you want to see it. Queries are used to display most objects on a list window. You can also decide what columns of data display on the list window, and how it will be sorted. Queries also have the ability to provide selection criteria. Selection criteria can be used to limit the records that display in a list window. Queries can also be used to improve performance by limiting the number of documents and columns that appear on a list window.

The query function in WebSphere Data Interchange Client enables you extensive control over the information that appears in any list window. Queries control both the number of documents that are listed and the number of fields that display. They are particularly useful after you have been using WebSphere Data Interchange for a while and the number of items on your list window is large.

A query is executed each time any list window is opened in WebSphere Data Interchange Client. You see the results of that query display on the list window of each tab. For example, when you open the Data Transformation Maps list window in the Mapping Functional Area, the default All query is often run. This query locates all Data Transformation maps and opens the results on the list window. The list window opened with this query lists all Data Transformation maps and sorts them by name.

You can change which query is executed to display the result on the list window. It can be changed to any available query that exists on the database. The available queries can include default queries provided by WebSphere Data Interchange and other queries that you or other persons have created. Click Properties on the list window toolbar to change which query is executed on an open list window.

A query can be created that is public, protected, or private. This function is only useful in installations where the WebSphere Data Interchange database is shared by several users. A public query is available to all users of the shared database. Any user can access the query and can alter or even delete the query. Protected queries are available to all users of the shared database; however, only the creator of the query can alter or delete it. Only the user that created the query can access private queries.

Chapter 3. The WebSphere Data Interchange mapping interface

WebSphere Data Interchange Client is designed to make setup, maintenance, and management of WebSphere Data Interchange Client components easier by using the Microsoft Windows graphical environment. Through the WebSphere Data Interchange Client interface you can perform the following tasks:

- Create, update, and manage the following WebSphere Data Interchange Client components:
 - Maps
 - XML document definitions
 - EDI Standards
 - Profiles, tables, and other supporting objects
 - Proprietary document definitions (data formats)
- Import Extensible Markup Language (XML) schemas and Document Type Definitions (DTDs) into WebSphere Data Interchange
- Set up and maintain database definitions
- Set preferences
- Work with the message log

This document focuses on the components of the WebSphere Data Interchange Client that are applicable to mapping. For all other components, see the *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01 for more information.

Main application window

The WebSphere Data Interchange Client main application window is used to manage all functions that can be performed in WebSphere Data Interchange Client. It contains a navigation bar and a menu. The navigator bar consists of a series of buttons that open the functional areas of WebSphere Data Interchange. A button is provided to open the help for WebSphere Data Interchange Client. A database selector is also provided. You select the database you want to work with before opening any functional area.

Use the menus to access all functions of the WebSphere Data Interchange Client. The menus that appear are dynamic, in other words the contents of each menu varies depending on whether a list window or editor is active.

Most mapping work is done in list windows and editors. A list window opens members of a single component. List windows can display by themselves or in a functional area window. A functional area window contains several list windows, each listing members from related components. The members selected for display in a list window are governed by the current query for that component. Most components in a functional area use a default query that results in all members of the component being selected and displayed in the list window. The default query opens only the most commonly referenced columns in the list window to improve performance.

Main application window

List windows can be opened individually without the involvement of a functional area. This occurs by selecting the Open Query List function on the File menu or by clicking buttons within various dialog or editors within the application. The Open Query List function opens the Query list window. Within this list window, you can select the component you want to work with along with the query to use to select members of the component to be displayed.

You can select alternate queries once the list window is opens. One or two other queries are provided for each component type by default. With these default queries, you can specify selection criteria while the other, if provided, lists most columns for each item within a component. You can create and use your own queries. Use of selection criteria is a useful way to create queries that list only a subset of the members of a component. It is recommended that you include only the columns that are really needed in a query. This improves performance, especially when listing a large number of members, or the database is not local, or if network performance is not optimal when accessing the database.

Use list windows by themselves or within a functional area to maintain the individual items of an object type. Items can be added, edited, renamed, deleted, and so on, using the list window.

You can have several functional area windows opened simultaneously. Also, the same functional area list window can be opened more than one time thus opening several list windows for the same component.

WebSphere Data Interchange Client can view multiple databases at the same time. You can work with development, test, and production database simultaneously. When opening a functional area, it opens using the database selected on the navigator bar. editors are used to view and update items in components. In most cases, they are also used to create the items in a component. Another way of creating items in a component is to import them into a database. There are a few item types that use a wizard dialog to collect the needed information and then create the item.

For more information of the navigation bar, menus, list windows and editors see the *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01 for more information.

Mapping functional area

The Mapping functional area is used to view and maintain maps and related items. You access the Mapping functional area by clicking Mapping on the WebSphere Data Interchange Client navigator bar. This opens a functional area window that contains several list windows. The following table describes the mapping components represented in each list window of the Mapping functional area. Figure 2 on page 20 shows the Mapping functional area.

Table 3. Mapping functional area components

Component	Description
Data Transformation maps	Data Transformation maps are used to transform a document in any format into a document in any format.
Validation maps	Validation maps provide additional validation for EDI documents.
Functional acknowledgement maps	Functional acknowledgement maps are used to create EDI functional acknowledgements. WebSphere Data Interchange Client provides a basic set of Functional Acknowledgement maps to produce common functional acknowledgements. You need to create your own or modify these only if you have special requirements.
Send maps	Send Maps are one of the two original WebSphere Data Interchange map types. This map converts a Data Format source document to an EDI Standard Transaction target document.
Receive maps	Receive Maps are one of the two original WebSphere Data Interchange map types. This map converts an EDI Standard Transaction source document to a Data Format target document.
Control strings	WebSphere Data Interchange does not use maps directly to translate or validate documents. Instead, map controls strings are used. Using WebSphere Data Interchange Client, maps are compiled into map control strings for use in translation and validation.
Global variables	Global variables are used with Data Transformation maps, validation maps, and Functional Acknowledgement maps. During translation, you can put data into a global variable. The data contained in a global variable is available to other translations.
Forward translation table Reverse translation table	Translation tables are used to change one value into another corresponding value during translation of a document.
Code lists	Code lists define a list of acceptable values that are used in maps to validate any source simple element or variable.

XML functional area

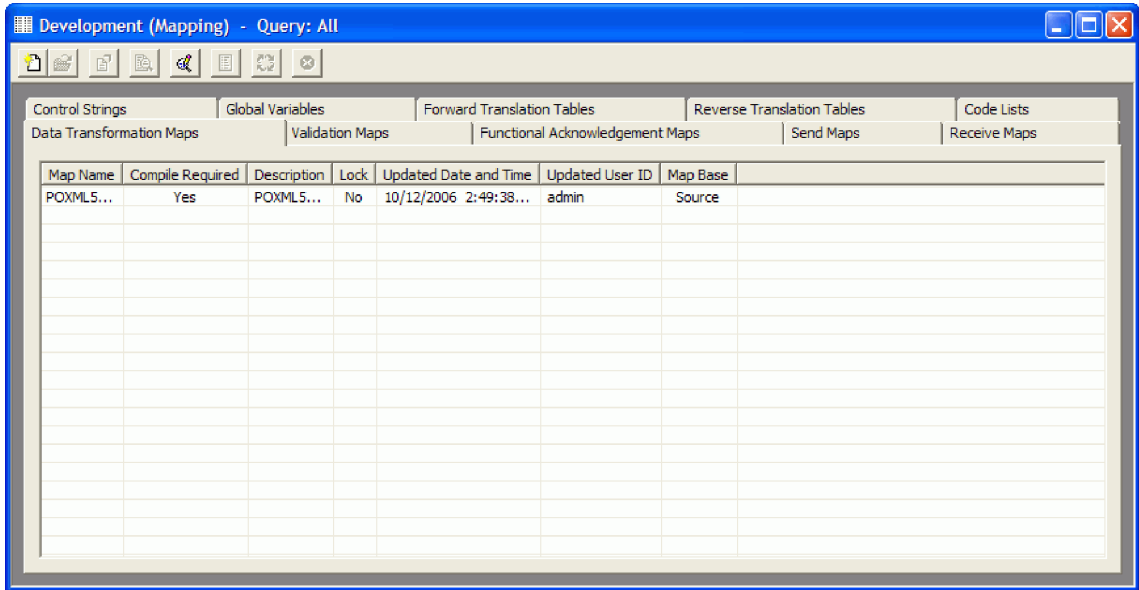


Figure 2. Mapping functional area

XML functional area

The XML functional area provides access to XML dictionaries, schema document definitions, DTD document definitions, and Namespace objects. You can access the XML functional area by clicking XML on the WebSphere Data Interchange Client navigator bar. This opens a functional area window that contains one list window for each XML related object type. Figure 3 on page 21 shows the XML functional area.

The XML functional area contains one list window for each of the following components:

- XML dictionaries v Schema document definitions
- DTD document definitions
- Namespace objects

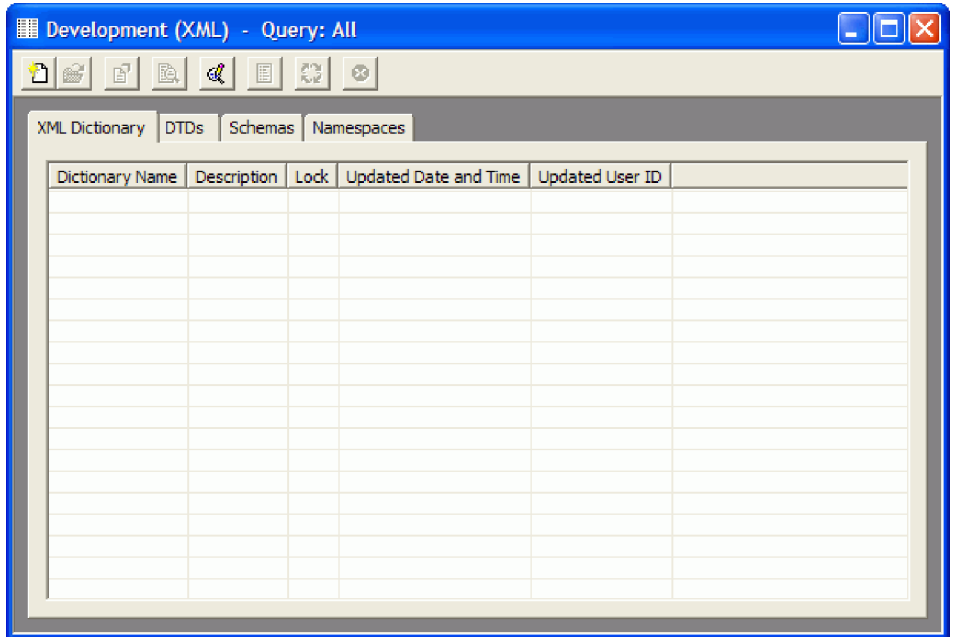


Figure 3. XML functional area

XML provides a readable, easy to use data format that trading partners use to exchange data between their computer applications. In essence, XML provides the building blocks for electronic versions of common business documents.

WebSphere Data Interchange can translate data from one document format into an XML document for transmission to a trading partner. Conversely, WebSphere Data Interchange can translate data in an XML document into another format.

In order to begin exchanging XML documents with a trading partner, you must select an XML schema or DTD that defines the layout of the document you want to send or receive. The schema or DTD must be imported into WebSphere Data Interchange Client.

EDI functional area

The EDI functional area is used to view and maintain EDI Standards. You can access the EDI functional area by clicking EDI on the WebSphere Data Interchange Client navigator bar. This opens a functional area window that contains several list windows. Each of these list windows opens a component that is part of the EDI functional area. Figure 4 on page 22 shows the EDI functional area.

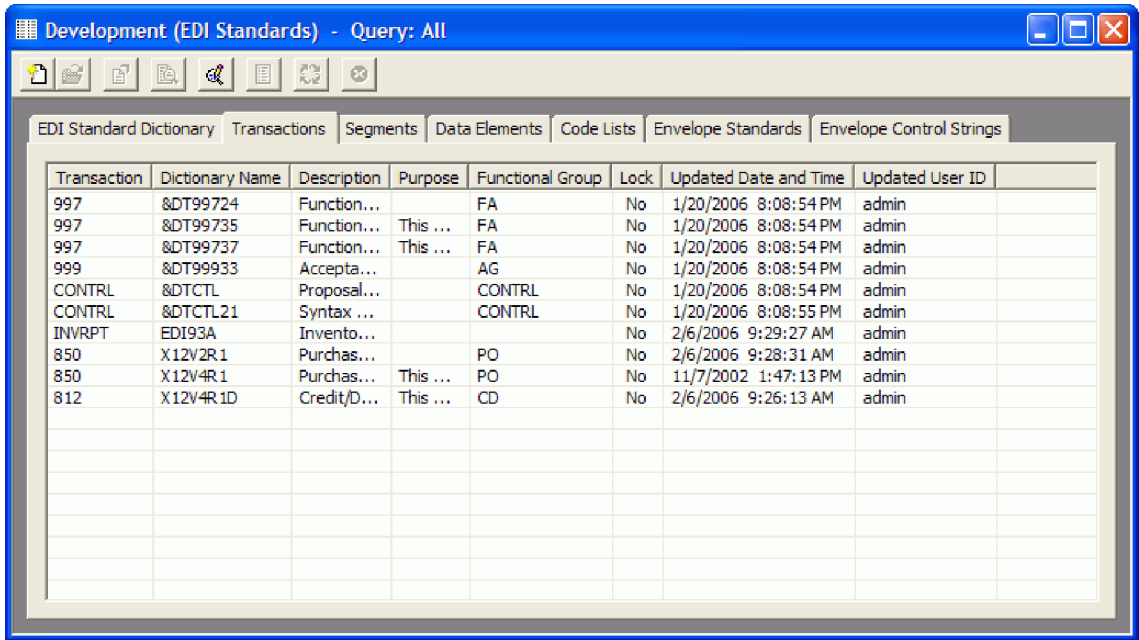
The EDI functional area contains one list window for each of the following components:

- EDI dictionary
- EDI document definitions

EDI functional area

- EDI segments
- EDI data elements
- Code lists
- Envelope Standards
- Envelope Control Strings

Note: Envelope Standards and Envelope Control Strings are used for send and Receive maps only. These do not apply to Data Transformation maps.



The screenshot shows a software window titled "Development (EDI Standards) - Query: All". The window has a menu bar with options: EDI Standard Dictionary, Transactions, Segments, Data Elements, Code Lists, Envelope Standards, and Envelope Control Strings. Below the menu bar is a table with the following columns: Transaction, Dictionary Name, Description, Purpose, Functional Group, Lock, Updated Date and Time, and Updated User ID. The table contains several rows of data, including entries for transactions 997, 999, CONTRL, INVRPT, 850, and 812.

Transaction	Dictionary Name	Description	Purpose	Functional Group	Lock	Updated Date and Time	Updated User ID
997	&DT99724	Function...		FA	No	1/20/2006 8:08:54 PM	admin
997	&DT99735	Function...	This ...	FA	No	1/20/2006 8:08:54 PM	admin
997	&DT99737	Function...	This ...	FA	No	1/20/2006 8:08:54 PM	admin
999	&DT99933	Accepta...		AG	No	1/20/2006 8:08:54 PM	admin
CONTRL	&DTCTL	Proposal...		CONTRL	No	1/20/2006 8:08:54 PM	admin
CONTRL	&DTCTL21	Syntax ...		CONTRL	No	1/20/2006 8:08:55 PM	admin
INVRPT	EDI93A	Invento...			No	2/6/2006 9:29:27 AM	admin
850	X12V2R1	Purchas...		PO	No	2/6/2006 9:28:31 AM	admin
850	X12V4R1	Purchas...	This ...	PO	No	11/7/2002 1:47:13 PM	admin
812	X12V4R1D	Credit/D...	This ...	CD	No	2/6/2006 9:26:13 AM	admin

Figure 4. EDI functional area

EDI Standards provide a common data format that trading partners use to exchange data between their computer applications. In essence, EDI Standards provide the building blocks for electronic versions of common business documents.

WebSphere Data Interchange can translate data from one document format into an EDI document. Conversely, WebSphere Data Interchange can translate data in an EDI document into another format. When you install WebSphere Data Interchange, you receive copies of EDI Standards currently approved by the primary standards organizations.

In order to begin exchanging EDI documents with a trading partner, you must select an EDI document definition that corresponds to the information you want to send or receive. Ideally, you and your trading partner can agree on an EDI document definition that requires no customization to meet your needs, but this is not always possible. If

you and your trading partner need to exchange specific information that is not included in existing EDI Standards, you can customize currently approved EDI Standards to fit your needs.

Note: When altering EDI Standards, you work in close partnership with your trading partners. If you customize EDI Standards without informing your trading partners of the changes, they might not be able to process the EDI documents you send. If you need to alter a component of an EDI Standard, it is recommended that you copy the EDI Standard and then alter the copy.

EDI functional area

Chapter 4. Data formats

The term *data format* defines the layout of your application data. It is a document definition. The word *data* refers to the information itself. The word *format* refers to the physical layout of information in the file, such as field names and lengths.

WebSphere Data Interchange requires a description of the data format for each business application that generates data for translation, or uses translated data. Application data must be described to WebSphere Data Interchange so that it can be used as either a source or target for translation.

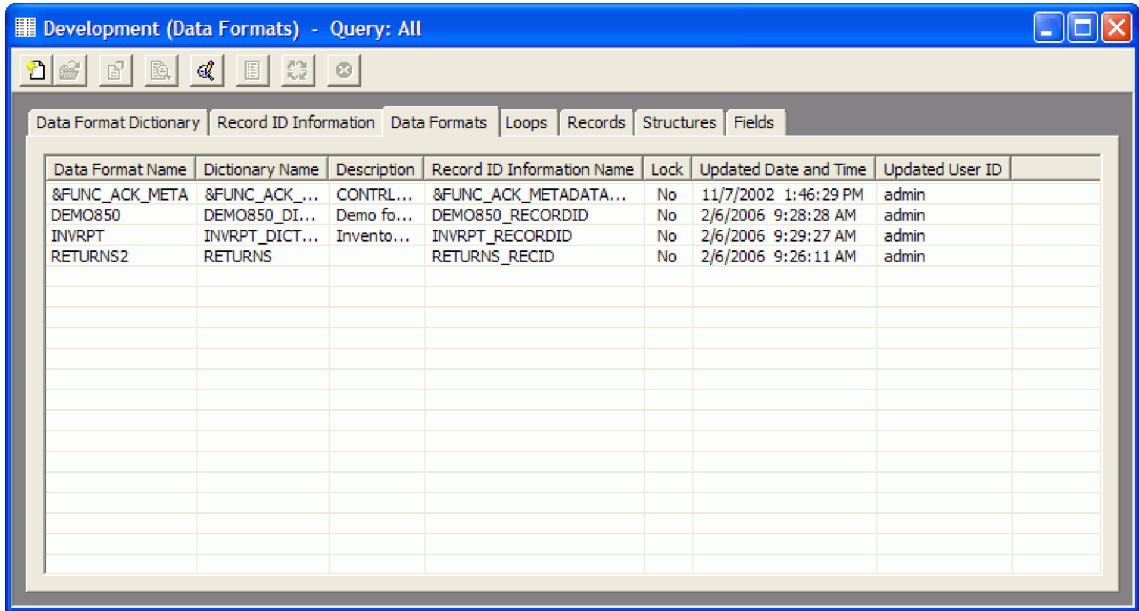
You can use the Data Format editor to describe the application's data to WebSphere Data Interchange. After you created a data format, you then create a map between the application's data format and the source or target document. Creating a data format is the first step in the mapping process if your map uses or creates data to be used by your application.

You usually need to create a data format for every unique business document that is used or created by WebSphere Data Interchange. A single data format can be mapped to multiple documents.

For example, If you use EDI or XML to exchange invoices, you need to create a data format for your invoicing system. This enables WebSphere Data Interchange to understand how your invoicing system structures an invoice. From that data format, you create a map to translate the application's data to either the 810 transaction in the X12 standard or the XML format that your trading partner uses. For details on mapping, Chapter 8, "Creating a map," on page 93.

This chapter assumes that you know the layout of the application data you are using.

Creating a data format



The screenshot shows a software window titled "Development (Data Formats) - Query: All". The window has a menu bar with icons for file operations and a tabbed interface. The active tab is "Data Format Dictionary". Below the tabs is a table with the following data:

Data Format Name	Dictionary Name	Description	Record ID Information Name	Lock	Updated Date and Time	Updated User ID
&FUNC_ACK_META	&FUNC_ACK_...	CONTRL...	&FUNC_ACK_METADATA...	No	11/7/2002 1:46:29 PM	admin
DEMO850	DEMO850_DI...	Demo fo...	DEMO850_RECORDID	No	2/6/2006 9:28:28 AM	admin
INVRPT	INVRPT_DICT...	Invento...	INVRPT_RECORDID	No	2/6/2006 9:29:27 AM	admin
RETURNS2	RETURNS		RETURNS_RECID	No	2/6/2006 9:26:11 AM	admin

Figure 5. Data formats functional area

Creating a data format

This section provides the basic steps you follow to create a data format for an application. The basic steps are:

1. Understand how your application data is structured.
Get a copy of your application's record layout and study the format. For more information about application data see 26
2. Fill out a data format work sheet.
This form helps you enter information about your data format into the data format editors. See the sample on page 31 and the worksheet example on page 31.
3. Use the data format editors to create the data format.
WebSphere Data Interchange has data format editors for Data Format Dictionaries, Record ID Info, Data Formats, Loops, Records, Structures, and Fields. See "Using the WebSphere Data Interchange data format editors" on page 33.

Each step is detailed in the sections that follow.

Understanding how your application data is structured

Understanding how application data is structured requires three steps:

1. Obtain a useful copy of your application data layout.
2. Optionally, structure that data so that WebSphere Data Interchange can use it.
3. Determine the data components used by your application.

Obtaining application data layout

First, you need to obtain a copy of each application's record layout. The record layout can come directly from the program code listings or any other documentation that shows the beginning and ending position of each field in the record. For each record, the information shows:

- Physical attributes of each field
- Content of each field
- Position of each field in the record
- Length of each field
- Data type of each field
- Relationship between the records

Structuring application data

The next step in understanding your application data is structuring application data in a format that WebSphere Data Interchange accepts. WebSphere Data Interchange can accept two types of data record formats: *raw data records* and *control and data records*.

Raw data: Each record in raw data format identifies itself by containing a unique record identifier (a record ID). Raw data can be either comma-separated or fixed-position.

- In fixed-position data, the identifier starts in the same position and extends for the same length in each record for send and Receive maps. For Data Transformation maps the record id can consist of several fields and vary by record.
- In comma-separated data, each value is separated by a comma. Comma-separated data is used only with Data Transformation maps. Raw data that is fixed-position can be used in Data Transformation maps, Send maps and Receive maps. The Record ID is actually a field in the record. As long as records contain identifiable record IDs, you can use application data without modification.

Figure 6 is an example of fixed-position data, several records of application data contain the record ID in the first three positions. In this example, HDR is the record ID of the header record, NAM is the name, and so on.

```
HDR0123456 092091C321
NAMSmithson, Patricia Jeanne
SSN555555555555
PRVCity Regional Clinic
PID05050505-X505
ADR555 Cedar Road
ADRAny City IL
ADR61001-1101
TOT1555.00
```

Figure 6. Application data with record identifier

Creating a data format

When your data has no record identifier for WebSphere Data Interchange to associate with each record, you have two choices. You can modify your application data to contain a record identifier or you can modify it to use control (C) and data (D) records.

Control and Data (C and D) Format: If no record ID clearly specifies the type of information contained in a record, that record structure can be indicated by using control and data records. Also use C and D records when you need to use multiple data formats in a single file or you need to use overrides offered in the C record that are not offered in raw data. You also can use this type of format to override fields within service segments (such as ISA, GS, UNB, and UNH).

There are a number of ways to structure C and D records. Table 4, for example, shows an example of a C and D record in which:

- A C or D is in the first column (byte)
- Record name is in the next 16 columns
- Application data starts in column 18

This example also shows the use of a Z record to indicate the end of the document. For more about Z records, see the *WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide*, SC34-6217-01.

Table 4. Sample Application Records with C and D Records

	1	2	3	4	5	6
	1234567890	1234567890	1234567890	1234567890	1234567890	1234567890
CSPSTT16				AFTSU09	IL	
DPOHDR		P0123456	092091C321			
DP0NOTE		INCOMPLETE INVOICE INFORMATION SLOWS REIMBURSEMENT				
DP0NOTE		PATIENTS WITH MULTIPLE CLAIMS NEED COMPLETE HISTORY				
DINVITEM		005500550055				
DITEMDESC		SURGICAL PROCEDURE				
DITEMDIAGN		CARPAL TUNNEL PAIN				
DINVITEM		005500550055				
DITEMDESC		SURGICAL PROCEDURE				
DITEMDIAGN		LIGAMENT INFLAMMATION				
DNAME		SMITHSON, PATRICIA JEANNE				
Z						
CSPSTT16				AFTSU09	IL	

Determining data components

You must determine the data components used by your application so that WebSphere Data Interchange can identify and process the data accurately. A *data component* is a grouping of related data fields, such as the field names that make up line items of a health-care claim or the ship-to address of a purchase order. WebSphere Data Interchange needs information about:

- Which fields in your application data form the various components WebSphere Data Interchange uses
- The order in which components occur
- The number of times each component occurs

That information identifies the data from your application. When you use the data format editors to create a data format for an application's data, you show relationships between the various components and show the number of times each component can occur in a transaction.

WebSphere Data Interchange uses four components to express the characteristics of the application's data for mapping:

- Fields
- Structures
- Records
- Loops

Fields: Fields are fundamental pieces of data, such as prices or item numbers or first names. Raw data that is fixed position can be used in Data Transformation maps, Send maps and Receive maps. In COBOL records, they are stored in a single variable.

Structures: A structure is a group of related data fields, which is probably unique to your company. When multiple fields always display together, you can designate the group as a structure and give it a structure name. Structures not only contain fields, but can also contain other structures.

For example, a purchase order line item contains price, quantity, and product ID fields. Those three fields always display together. Further, they are used not only on purchase orders but also on invoices. So you can create a structure called a line-item structure that consists of price, quantity, and product ID fields. A purchase order line item, then, can consist of a line item structure plus a requested ship date field, while an invoice line item can consist of a line item structure plus a requested payment date field. You can even repeat that structure within a purchase order line item record.

Note: Structures cannot be used for comma-separated data.

Records: A record is a set of related fields as they are defined in an application's data. Every record in your application must be defined in your data format, assuming you want to map all records. Records contain fields and structures. An example of a record that identifies a patient in a hospital's claims-management system is described in the following example.

You are a health care provider and you want to send your claim information in a single health-care claims EDI standard transaction to the insurer or payer of the claim. You submit claims each day for all patients with the same insurer or payer. You would look at your application's data and determine which records are required to send the data for a patient claim, say a patient record followed by claim line-item records. The patient is identified as R20, and the claim information is identified as R42 and R73, as shown in Table 5.

Data formats: records

Table 5. Sample Record

R20	01623792001	ALBERT	MICKEY
XF07261925S31PO BOX 11			
R42	0101623792001	C019200002510{0005{21000000150{0001{155000002500000A	
R42	0201623792001	PR1	00006100{0000{ 00000000{0000{
000000000000			
R73	01623792001	48521PATIENT IS RECOVERING	
FROM SURGERY AS NEEDED			
R20	01623792001	LOUISE	MOONIE
XF07261925S31PO BOX 11			
R42	0302623792001	OA1	00006100{0002{8900000100{0206{115000005000000E
R73	01623792001	48521PATIENT SHOULD RETURN	
IN 2 MONTHS FOR LAB RESULT			

In this example, there are multiple patient records and each patient record contains that patient's claim information. Because this group of records is related and repeats, you would create a loop consisting of the patient record and the repeating claim line-item records. Because loops can repeat, and the health care claims EDI standard transaction defines a repeating patient claim loop, you can send claims for multiple patients in a single health care claims EDI standard transaction to a single insurer or payer.

Loops: A loop is a group of records that repeat. Loops can occur within other loops; this is referred to as a nested loop. Loops are used for repeating such things as line items, as in the following example:

You want to include claims for multiple patients in a single health-care claims transaction. You would look at your application's data and determine which records are required to send the data for a patient claim, say a patient record followed by claim line-item records.

To pay multiple patients in the same transaction you would create a loop consisting of a patient record and a repeating claim line-item record. Because loops can repeat, you can send claims for multiple patients by repeating the patient claim loop for every patient.

When deciding how best to structure your application's data into data components, keep the following component hierarchy in mind:

- Loops can contain other loops or records.
- Records can contain structures or fields.
- Structures can contain other structures or fields.
- Fields are fundamental units of data.

The data format worksheet example in the following section helps you to determine how to structure your application data into data format components.

Filling out a data format worksheet

A data format worksheet can make it easier to decide how to structure application data. This example shows a purchase order record definition and how its data format worksheet looks.

```

01 SHIPTO
  05 RECORD-IDENTIFIER
    10 PURCHASE-ORDER-NUMBER      PIC X(08).
    10 RECORD-ID                   PIC X(02).
  05 COMPANY-NAME                  PIC X(30).
  05 COMPANY-DUNS                  PIC X(10).
  05 COMPANY-ADDRESS
    10 STREET                      PIC X(30).
    10 CITY                       PIC X(15).
    10 STATE                       PIC X(02).
    10 ZIPCODE                     PIC 9(09).
  05 COMPANY-PHONE                PIC 9(10).
01 DETAIL
  05 RECORD-IDENTIFIER
    10 PURCHASE-ORDER-NUMBER      PIC X(08).
    10 RECORD-ID                   PIC X(02).
  05 ITEM-NUMBER                   PIC X(10).
  05 ORDER-QUANTITY               PIC 9(09).
  05 ORDER-UNITS                  PIC X(01).
  05 UNIT-PRICE                   PIC 9(09)V99.
  05 UNIT-DISCOUNT                PIC 9(09)V99.
  05 EXTENSION                    PIC 9(09)V99.
*THERE ARE UP TO 3 DETAIL DESCRIPTION RECORDS PER DETAIL RECORD.
01 DETAIL-DESCRIPTION.
  05 RECORD-IDENTIFIER
    10 PURCHASE-ORDER-NUMBER      PIC X(08).
    10 RECORD-ID                   PIC X(02).
    10 DESCRIPTION
PIC X(30).
01 TOTAL
  05 RECORD-IDENTIFIER
    10 PURCHASE-ORDER-NUMBER      PIC X(08).
    10 RECORD-ID                   PIC X(02).
  05 TOTAL-AMOUNT                 PIC 9(09)V99.

```

Figure 7. Sample Working Storage Record Definition of a Purchase Order

Table 6. Data Format Component Relationship Worksheet Example

Parent Type	Parent Name	Child Type	Child Name	Max Use or Occurs	Record ID (REC Only)	Field Data Type (FIELD Only)	Field Length (FIELD Only)
data format	SAMPLE-PO	REC	SHIP-TO	1	SH		
		LOOP	DETAIL-LOOP	1000			
		REC	TOTAL	1	TT		

Data format worksheet

Table 6. Data Format Component Relationship Worksheet Example (continued)

Parent Type	Parent Name	Child Type	Child Name	Max Use or Occurs	Record ID (REC Only)	Field Data Type (FIELD Only)	Field Length (FIELD Only)
REC	SHIP-TO	STRUCT	RECORD-IDENTIFIER	1			
		FIELD	COMPANY-NAME			CH	30
		FIELD	COMPANY-DUNS			CH	10
		STRUCT	COMPANY-ADDRESS	1			
		FIELD	COMPANY-PHONE			NO	10
STRUCT	RECORD-IDENTIFIER	FIELD	PURCHASE-ORDER-NUMBER			CH	8
		FIELD	RECORD-ID			CH	2
	COMPANY-ADDRESS	FIELD	STREET			CH	30
		FIELD	CITY			CH	15
		FIELD	STATE			CH	2
		FIELD	ZIPCODE			NO	9
LOOP	DETAIL-LOOP	REC	DETAIL	1	DE		
		REC	DETAIL-DESCRIP..	3	DD		
REC	DETAIL	STRUCT	RECORD-IDENTIFIER	1			
		FIELD	ITEM-NUMBER			CH	10
		FIELD	ORDER-QUANTITY			NO	9
		FIELD	ORDER-UNITS			CH	1
		FIELD	UNIT-PRICE			N2	11
		FIELD	UNIT-DISCOUNT			N2	11
		FIELD	EXTENTION			N2	11
	DETAIL-DESCRIP..	STRUCT	RECORD-IDENTIFIER	1			
		FIELD	DESCRIP..			CH	30
	TOTAL	TOTAL	STRUCT	RECORD-IDENTIFIER	1		
FIELD			TOTAL-AMOUNT			N2	11

Table 7. Data Format Component Relationship Worksheet

Parent Type	Parent Name	Child Type	Child Name	Max Use or Occurs	Record ID (REC Only)	Field Data Type (FIELD Only)	Field Length (FIELD Only)

Using the WebSphere Data Interchange data format editors

When you have filled out the data format worksheet, you are ready to create the data format on WebSphere Data Interchange Client. To create the data format and its components, use the data format editors, as described in this and following sections.

You use the data format editors to create the various components that make up a data format. In using the editors to create a data format, first decide whether to work from

Data format editors

the top down or from the bottom up. If you work from the top down, after you create a data format dictionary, you create the larger components first, then work down to the smaller. If you work from the bottom up, after you create a data format dictionary, you create smaller components first, and then work up to larger.

Whichever way you choose to work, you must create a data format dictionary as your first step. You cannot create components for a nonexistent dictionary and store them elsewhere until the dictionary is created. When you have created the dictionary, you can create the smaller components such as structures and fields before creating the larger components. In practice, it is easier to create the larger components, such as loops and records first, and then work down to the smaller components until your data format is complete.

This section:

- Provides a generic description of the procedures for using the data format editors.
- Steps through the process of using the editors to create a new data format.

Accessing data format editors

You use the data format list window to gain access to the data format component editors. Each tab contains a list of components. Click the tab corresponding to the component you wish to work with. From that list, you can select the specific component you wish to work with and open its editor by clicking the component and then clicking Open.

To access a data format editor:

1. Click Data Format on the WebSphere Data Interchange Navigator bar.

The data format list window opens.

2. Click the tab of the data format component you wish to work with.

The list window for that component opens.

This list window opens a list of existing components of the type you selected. Each row contains a single component; each column contains data stored in the component. Information in the columns displays in fields in the respective editor. The list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the page to scroll to the right or left. To alter the columns that display on the page, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01

3. To view an existing component or to add or change its information, double-click the first column of the item.

The editor opens, with the **General** tab or **Details** tab in front, depending on which component type you opened. You add information or make changes to the data format component through its tabs, as described in the following sections.

Note: All of the component editors have a **General** tab and a **Comments** tab. The Data Formats, Loops, Records, and Structures editors also contain a **Details**

tab for setting up the specifications for those components. The Data Formats editor has two additional tabs, an **Overview** tab and a **Raw Data** tab. The **Overview** tab provides a visual representation of the entire data format. The **Raw Data** tab indicates which fields contain important information that can be used in a translation by the WebSphere Data Interchange Server. The information is used only when the Data Format is Raw Data format.

The following sections contain detailed procedures for creating data format components. For information about viewing, copying, editing, renaming, deleting, and printing data format components, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. For information about exporting data format components, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

The data format component editors are described in the following sections in the order in which you use them when creating a data format from scratch following a top-down approach.

Creating a data format dictionary

A data format dictionary is a name for a group of other components. The following instructions detail creating a new data format dictionary. For information about viewing, copying, editing, renaming, deleting, and printing data format dictionaries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. For information about exporting data format dictionaries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new data format dictionary when you set up your first data format for a particular application. You can then reuse data format components for that application when you create subsequent data formats if they are created within that dictionary. You can also import COBOL copybooks using the data format dictionary editor.

1. Select the **Data Format Dictionary** tab, click New on the list window tool bar.

The data format dictionary editor opens with the **General** tab in front and the fields blank.
2. Type a name in the Dictionary Name field.

Notes:

 - a. The name is displayed in uppercase letters.
 - b. You cannot type spaces within the name.
 - c. The name can be up to 30 characters long and can contain alphanumeric characters and any of the special characters “!@#%&*()_-=':;<,>.”.
3. Enter a description of the data format dictionary in the Description field. This field is optional.
4. Click Save on the tool bar to save the dictionary.

After you have saved your dictionary, the Dictionary Name becomes read-only.

Note: When you save the dictionary, the Data Formats, Loops, Records, Structures, and Fields buttons become available in the List Of group box at

Data format dictionary editor

the bottom of the editor. Click those buttons to open list windows that contain the components associated with this dictionary. When you first create a dictionary, the lists are empty.

Importing a COBOL copybook

Importing uses a COBOL source file to create WebSphere Data Interchange fields, structures, and records, and then save them in this data format dictionary. The data format records can be assigned to an existing data format or to a new data format.

1. Use the procedure described in “Creating a data format dictionary” on page 35 to create a dictionary to store the new records and fields. The dictionary must be saved before you proceed. After you save your dictionary, the Dictionary Name becomes read-only and the Import COBOL Copybook button is enabled.
2. Click Import COBOL Copybook.

The Select COBOL Copybook File window opens.

3. Locate and select the appropriate file. Click Open.

Import of the COBOL copybook begins. The Status window opens. Messages relating to the import of the copybook are shown in this window. The following occurs during import of the COBOL copybook:

- A Data Format Record is created or updated for each COBOL record description (01 level).
- A Data Format Structure is created or updated for each COBOL group item.
- A Data Format Field is created or updated for each COBOL elementary item.
- A Code List is created or updated for each COBOL elementary item condition name entry (88 level).

Note: The components created or updated by the import of the copybook have not been saved to the System. Saving the created or updated components comes later in this process. The Import COBOL Copybook button is disabled until a save is performed. Additional COBOL copybooks can be imported once a save is performed.

4. Click Close when the status window indicates the import is complete.

The path and file name of the copybook file display in the Copybook File field. The Data Format Name and Record ID Information lists as well as the corresponding New buttons are enabled.

Note: If errors occurred during the import of the copybook, evaluate and correct the errors and then try to import the copybook again.

5. Select an existing data format in this dictionary from the list, or click New. If this is a new dictionary, there are no data formats in the drop down list.
 - If a valid data format is selected, the Record ID Info and its New button are disabled. Go to step 7 on page 37.
 - If you click New, the Data Formats editor opens. See “Creating a data format” on page 40. Be sure you save the new data format before returning to the data format dictionary editor. Go to step 6 on page 37.

6. Select the existing Record ID information object from the list, or click New. This is a mandatory field. The record ID information object provides information about how the record is identified to WebSphere Data Interchange.

If you click New, the data format record ID information editor opens. See “Creating the data format record ID information” on page 38. Be sure you save the new record ID information before returning to the data format dictionary editor.

7. Check the Convert 88s to Codelists box, if you want to convert 88 entries to code lists. One code list is created for each set of 88-level statements that are associated with the COBOL data name. The names assigned to the code lists are determined by values in the Prefix and Starting Number fields. Checking this box enables the Prefix and Starting Number fields.

Prefix The prefix is a value up to five characters in length that is used as the first part of the name for all created Code Lists. The default is the first five characters of the name of the Data Format dictionary.

Starting Number

This field is a one to three digit number. The first Code List created by importing the COBOL copybook has this value appended to the specified prefix to form the name of the Code List. WebSphere Data Interchange increments this value for each subsequent Code List created and uses the increased value and the prefix as the name of the Code List. This value default is 1 and has a range of 0 to 999.

8. Click Save on the tool bar to save the COBOL objects in the dictionary.

After the import process is complete, you must update any Data Format created as part of the import of the copybook. In addition, any existing Data Format that is changed by the import of the COBOL copybook can also require updating. To update the Data Format:

1. Open the Data Formats list window by selecting the **Data Formats** tab in the Data Formats Functional Area.
2. Select the Data Format involved in the import of the COBOL copybook and click **Open** on the list window toolbar.
3. Select the **General** tab on the Data Format editor and complete the fields. For more information about the Data Format editor fields see “Creating a data format” on page 40.
4. If the copybook is using Raw Data format, select the **Raw Data** tab on the Data Format editor and complete the fields. For more information about the Data Format editor fields see “Creating a data format” on page 40.
5. Click Save to save your changes. WebSphere Data Interchange saves the changes to the System.
6. Close the Data Format editor and the Data Formats list window.

If you are using Raw Data format, you must do the following before using the Data Format:

1. Open the Data Format Records list window by selecting the **Records** tab in the Data Formats Functional Area.

Data format dictionary editor

2. Select all of the new Records and click Open on the list window toolbar. This opens the Data Format Record editor displaying the first selected Record.
3. Change the record ID to the required value.
4. Click Save on the editor toolbar to save the change. WebSphere Data Interchange saves the change to the System.
5. Click Next on the editor toolbar to navigate to the next record.
6. Repeat steps 3 through 5 to change the record ID on each Record created during the import of the COBOL copybook process.

Note: When using a Raw Data format, each Record in a document must have a unique record ID to determine what the record is. WebSphere Data Interchange is not able to determine the proper record ID value from the COBOL copybook.

7. Close the Data Format Record editor and the Data Format Records list window.

Understanding COBOL copybook REDEFINES and SYNC clauses

When a REDEFINES clause is encountered in a COBOL record description at a level other than the 01 level, the Data Format Structures or Data Format Fields created are included in the Data Format Dictionary but not in the current Data Format Record.

These Data Format Structures or Data Format Fields are attached to a Record that WebSphere Data Interchange creates called DATAINTERCHANGE-REDEFINES. After the Data Format Dictionary is saved, you can use these Data Format Structures or Data Format Fields in your Data Format.

When a SYNC clause is encountered in a COBOL record description, WebSphere Data Interchange inserts filler entries in the Data Format Structure or Record before the item with the SYNC clause to ensure it lands on the appropriate boundary. Use caution when reusing these Data Format Structures. Use of the SYNC clause in combination with the OCCURS clause or on group items is not supported.

Creating the data format record ID information

The data format record ID information defines records in application data. The definition indicates whether the data format uses raw data format records or C and D format records. When your records are structured using raw data format, the data format record ID information definition also specifies the location and length of the Record ID.

Unlike data format dictionaries and other components, record ID information definitions are global for a system; they are not tied to a specific data format dictionary but can be used in any data format dictionary in the system. If your company structures all application record ID information in the same way or always uses C and D records, then you only need to create one record ID information definition for use with any data format.

Create a new data format record ID information definition when you set up your first data format or when you set up a data format whose record IDs are structured differently than your existing data formats. You can use the same data format record ID information definition for any data format whose record IDs have the same structure or that are set up as C and D records.

The following procedure consists of instructions for creating a new data format record ID information definition. For information about viewing, copying, editing, renaming, deleting, and printing data format record ID info profiles, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

1. Select the **Record ID Information** tab and click New on the list window tool bar.
The data format record ID Information editor opens with the **General** tab in front.
2. Type a name in the Name field.

Notes:

- a. The name is displayed in uppercase letters.
 - b. You cannot type spaces within the name.
 - c. The name can be up to 30 characters long.
3. Enter a description of the data format record ID information definition in the Description field. This field is optional.
 4. Select the record format, raw data or C and D format records.
 - If your application uses raw data format select Raw Data and go to step 5.
 - If your application uses C and D format select C and D Records and go to step 9.

Note: When you select C and D Records the Record ID Information section of the page automatically fills in and cannot be edited.

5. Type the position of the first character of the Record ID in the Position field.
6. Select the Position Type. The Position Type specifies whether the number specified for the position of the record identifier is based on character, byte, or field.
 - Character
The character position type specifies that the location of the record identifier is based on the number of characters from the beginning of the record. The position and length of a field are multiplied by the character size specified in the Data Format to determine the byte offset and length of the field.
 - Byte
The byte position type specifies that the location of the record identifier in a record is based on the number of bytes from the beginning of the Record.
 - Field
The field position type specifies that the location of the record identifier in a record is based on the number of fields from the beginning of the record. For example, if Position were set to 3, this specifies that the record identifier is the third field of the delimited field Record, regardless of the byte or character offset.
This is only useful if the record identifier is used for a delimited field Data Format.
7. Type the length of the Record ID in the Length field. The default is 1 and the maximum is 64.
8. Select the Data Type from the list. Data Types are listed in Table 9 on page 54.
9. Click Save on the tool bar to save the profile.
After you have saved your record ID information, the Record ID Info and field becomes read-only.

Data format record ID information editor

To find other data format components that use the current data format record ID information definition:

1. Select the **General** tab page and click Where Used.
A list window opens containing **Data Formats**, **Loops**, and **Records** tabs.
2. Click the tab of each data format component to view a list of components containing the current record ID information definition.

You can open any component by double-clicking it.

An empty list window means that the data format record ID information definition is not used in any data format component of the list window's type.

Creating a data format

The data format editor defines and structures the components that make up a data format. A data format is a document definition. It defines the layout of your application's data. From the data format editor, you name the data format, identify properties related to the data format, and create and edit the associations to loops and records. It also gives you a visual of the data format from which you can directly navigate to each of its component editors.

The data format editor contains five tabs:

- The **General** tab contains fields for you to set the name of the data format, select a dictionary and properties of the data format.
- The **Details** tab contains fields for you to add or change loops and records associated with the selected data format and edit information about the association.
- The **Overview** tab displays a representation of the entire data format.
- The **Raw Data** tab contains fields for you to enter and change information specific to raw data format translation.
- The **Comments** tab contains a field for you to type comments about the selected data format.

The editor also contains a Where Used button, which lists all send and Receive maps that use the current data format.

Create a new data format when you need to define the new layout of application data that is used in translation.

The following instructions are for creating a new data format. For information about viewing, copying, editing, renaming, deleting, and printing data formats, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. For information about exporting data formats, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

1. Select the Data Formats tab and click New from list window tool bar. The Data Formats editor opens with the General tab showing.
2. Enter the name of the Data Format.

Notes:

- a. The name is displayed in uppercase letters.
 - b. You cannot type spaces within the name.
 - c. The name can be up to 16 characters long and can contain alphanumeric characters and any of the special characters “!@#%&*()_+ =':;<,>.>?”. It is suggested that the name not start with an ampersand (“&”) or dollar sign (“\$”). Data Formats provided by WebSphere Data Interchange begin with either of these two characters. If you use generic Send Map Usages, this name is limited to 8 characters.
3. Select the name of the Data Format Dictionary. This association enables the data format to utilize all Loops, Records, Structures, and Fields in the dictionary.
 4. Enter a description for the data format. This field is optional.
 5. Select the Record ID Information object. The Record ID Information object indicates whether the Data Format is in the C and D records format or the Raw Data format. It also indicates the position and length of the field that contains the record identifier.
 6. Select the Record delimiter.
 - If you select an option other than No delimiter, the Field Format section of the tab becomes available. Proceed with step 8.
 - If you select No Delimiter, proceed with step 10.
 7. Select whether the data is text only.

The Text only check box indicates that the source or target data contains only text characters (no binary data). Using this flag reduces processing time with the Data Transformation logical message adapter (LMA) and WebSphere Data Interchange MB message broker processors.
 8. Select the Field format.
 - For Data Transformation Map data formats, the records can contain either fixed length fields or delimited fields.
 - For Send and Receive Maps the records can contain only fixed length fields.
 9. If you select Delimited, then you must specify the Delimiter Information. This information specifies either a special character or text string that specifies columns within a data record.
 10. If you are using a code page other than the system default code page when reading or writing to a file using this Data Format, select the alternate code page. This field is ignored when the Data Format is used in Send Maps or Receive Maps.
 11. Specify the size of the characters in the document defined by this data format. Characters are one or two bytes in length, depending on the code page used.
 12. Specify the default output destination for documents created using the Data Format in the Document Destination Type and Document Destination Name fields. This information is optional.
 13. Click Save.
 14. Select the Details tab. Use this tab to identify the first level of Loops and Records that are contained in the Data Format. See the help topic on that tab for additional information.

Creating a data format

There are two ways to approach providing information for the Details tab:

- One method is to create the Loops and Records that are part of the first level of the Data Format by typing the new Loops and Records into the Details tab. After the Data Format has been saved, you need to open the Data Format Loop editor and Data Format Record editor for each Loop and Record created. In the editor complete the definition for each Loop and Record. This in turn requires you to create Structures and Fields that are a part of your Data Format.
 - Another method for creating a Data Format is to first create all of the Fields that are part of the Data Format. Then create all of the Structures that are part of the Data Format. Follow this up by creating all of the Records that are part of the Data Format. Create the Loops that are a part of the Data Format. Finally, create the Data Format. In this scenario, you use the Details tab of the Data Format editor to select the existing Loops and Records that are a part of the first level of the Data Format.
15. Select the Overview tab. This tab displays a graphical representation of your Data Format. The information about this tab is refreshed each time the Data Format is saved.

If you are using Raw Data format, you can right click Records and Fields on this tab and use functions on the popup menu to set certain fields in the Raw Data tab.
 16. If you are using Raw Data format (as opposed to C and D records format), set values on the **Raw Data** tab. You can set the values using a right click popup menu on the **Overview** tab or you can click the **Raw Data** tab.
 17. On the Raw Data tab:
 - a. Identify either the first or last record within the Data Format. Records you have defined in a Data Format do not have to be in the order they are defined to the Data Format, so one of these fields must be filled in when using Raw Data format so WebSphere Data Interchange can determine where each document begins and ends.
 - b. Identify the Field that contains the internal trading partner ID if the Data Format contains such a field. The internal trading partner ID is a value you use to identify your trading partner.
 - c. Identify the Fields that contains the interchange qualifier and interchange ID for the sending trading partner if the Data Format contains these fields. These fields you use to identify your trading partner.
 - d. Identify the Fields that contains the interchange qualifier and interchange ID for the receiving trading partner if the Data Format contains these fields. These fields you use to identify your trading partner.
 - e. Complete any other of the fields on the **Raw Data** tab as needed.
 18. Select the Comments tab and add any comments you have about the Data Format.
 19. Click Save the editor toolbar. WebSphere Data Interchange Client saves the new Data Format to the System.
 20. Close the editor when you are done.

Notes:

1. Complete the **General** tab before working on the **Details** tab. This ensures that all information is available when filling in the **Details** tab. Changing the name of the

Data Format, the Dictionary the Data Format belongs to, or the Record ID Information object used by the Data Format on the **General** tab causes all information about the **Details** tab to be reset. This is not an issue with existing Data Formats.

2. The **General** tab contains a Where Used button which lists the maps that reference the Data Format. Click the button and a list window opens containing tabs for each map. Select the specific tab to see the list of maps of that type that reference the current Data Format.
3. The name of a Data Format must be unique regardless of the Data Format Dictionary.

To see which maps use the current data format :

1. Select the **General** tab page and click Where Used.

A list window opens containing **Data Transformation Maps, Functional Acknowledgement Maps, Send Maps, and Receive Maps** tabs.

2. Click the tab of each map to view a list of maps using the current data format.

You can open any component by double-clicking it.

An empty list window means that the data format is not used in any map of the list window's type.

Creating loops

Use the data format loop editor to enter new loops into a data format dictionary or to edit existing loops. Loops are entered into a data format using the data format editor when you create or edit a data format. From the data format loop editor, you can create and edit the loop's associations with loops and records, as loops can contain loops and records.

The data format loop editor contains three tabs:

- The **General** tab contains fields for you to name the loop, select its dictionary and set loop properties.
- The **Details** tab contains fields for you to add or change loops and records associated with the selected loop and to edit the information about the association.
- **Comments** tab contain a field for you to enter any comments about the selected loop.

The **General** tab has a Where Used button. This button locates other data format components that use the currently selected loop.

The following instructions are for creating a new loop. For information about viewing, copying, editing, renaming, deleting, and printing loops, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new loop when your application data has two or more records that are grouped together and each group can occur more than once.

1. Select the **Loops** tab and click New on the list window tool bar. The data format loop editor opens with the **General** tab in front.

Data format loop editor

2. Type a name in the Loop Name field.

Notes:

- a. The name is displayed in uppercase letters.
 - b. You cannot type spaces within the name.
 - c. The name of a Data Format Loop must be unique only within the Data Format Dictionary in which the Loop belongs. However, it must be unique against all Data Format Loops, Data Format Records, Data Format Structures, and Data Format Fields within that same dictionary.
 - d. The name can be up to 30 characters long and can contain alphanumeric characters and any of the special characters “!@#%&*()_+=':;<,.>.”.
3. Select the data format dictionary in which you want the loop to occur using the Dictionary Name list. The Data Format Loop utilizes all Loops and Data Format Records defined within the same dictionary.
 4. Enter a complete description of the loop in the Description field. This field is optional.
 5. Select the **Details** tab to enter information about the loops and records contained in this loop.
 6. Click New to begin defining records and loops.
 7. Enter the position in the Position field. This field default is 1 and must be unique for each record and loop defined.
 8. Select the Type list. If this is the first entry in the loop, it must be a record.

Note: Loops can contain both loops and records, but must begin with a record. The value of Maximum Repeat for the first record must be 1.

9. Either select the name of the loop or record name from the Name list or type in a name to create a new loop or record. The list contains loops and records that are in the same dictionary and use the same record ID information definition as this loop.
 - If you type in a new name for the record or loop and click Insert, the Object Information section of the page becomes available. Go to step 10.
 - If you select an existing loop or record, the object information is filled in with that objects specific information. Go to step 12.
10. Enter a detailed description of the record or loop in the Description field.
11. If this is a record, enter a Record Identifier. The record identifier must be unique to the record within the Data Format.

Note: If you are using C and D records format, you do not need to specify a record identifier. Records are identified by the record name in C and D records format.

12. Specify the number of times to repeat the record or loop by either:
 - Entering a number between 1 and 32766 in the Maximum Repeat field.
 - Selecting the Unlimited Repeat check box.
13. At this point you have two options:

- To add the record or loop and continue adding records, click Insert and return to step 7 on page 44.
 - To add the record or loop and close the Data Format Loop Detail editor, click OK. The record or loop is added to the list window and the Data Format Loop Detail editor closes.
14. Click save on the editor tool bar.

To find other data format components that use the current data format loop:

1. Select the **General** tab and click Where Used.
A list window containing **Data Formats** and **Loops** tabs opens.
2. Click the tab of each data format component to view a list of components containing the current loop.
You can open any item in the list by double-clicking it.
An empty list window means that the loop is not used in any other data format component of the list window's type.

Creating a data format record

Use the data format record editor to enter new records into a data format dictionary or to edit existing records. From the data format record editor, you create and edit the record's associations with fields and structures, as records can contain fields and structures.

The data formats record editor contains three tabs:

- The **General** tab contains fields for you to name the record, select its dictionary, and set properties.
- The **Details** tab contains fields for you to add or change fields and structures associated with the selected record and to edit the information about the association.
- The **Comments** tab contain a field for you to type any comments about the selected record.

The editor also contains a Where Used button, which lists other data format components that use the currently selected record.

The following instructions are for creating a new record. For information about viewing and printing records, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new record when your application data layout requires one. Records can contain fields and structures.

1. Select the **Records** tab and click New on the list window tool bar. The Data Format Record editor opens with the **General** tab in front.
2. Type a name in the Record Name field.

Notes:

- a. The name is displayed in uppercase letters.
- b. You cannot type spaces within the name.

Data format record editor

- c. The name can be up to thirty characters long.
3. Select the data format dictionary in which you want the record to occur using the Dictionary Name list. Once you select a dictionary, the Record Identification section of the tab becomes available.
4. Enter a description of the record. This field is optional.
5. Select the appropriate radios button for how you want to identify the record.
 - Select Default to use the record ID specified in the Data Format and go to step 6.
 - Select Use a Specific Record ID Information Object if you wish to use an object other than the default and go to 6.
 - Select the Indicate the Record ID Using Fields field to select fields to use for the record ID and go to 8.
6. Type the value of the Record ID in the Record Identifier field. This is a required field.
 - If you selected Default in step 5 go to step 20 on page 47.
 - If you selected Use a Specific Record ID Information Object in step 5 go to step 7.
7. Select the Record ID Information Object to use from the list.

The area at the bottom of the pages updates to reflect the definition of the record selected. Verify this is the correct record and go to step 20 on page 47.
8. Click the **Details** tab to enter information about the fields and structures contained in this record.
9. Click New to open the Data Format Record Detail editor.
10. Enter the position in the Position field. This field default is 1 and must be unique for each field and structure defined.
11. Select Field in the Type list.
12. Select the name of the field from the Name list or type in a name to create a new field. The list contains fields that are in the same dictionary and use the same record ID information definition as this fields.
13. If the field contains the record ID, select the Field Contains the Record ID check box. The Record Identifier field becomes available. Specify the record identifier.
14. Continue defining the field.
 - If you specified a new name for the field and click Insert, the Object Information section of the page becomes available. Go to step 15.
 - If you select an existing loop or record, the object information is filled in with that objects specific information. Go to step 18.
15. Enter a detailed description of the record or loop in the Description field. This field is optional.
16. Select the data type from the Data Type list. For definitions of WebSphere Data Interchange data types, see Table 9 on page 54.
17. Enter the length of the field in the Field Length field. The field length value is a number between 1 and 32767.
18. At this point you have two options:

- To add the field to the list and continue adding fields, click Insert and return to step 10 on page 46.
 - To field and close the Data Format Record Detail editor, click OK. The field is added to the list window and the Data Format Record Detail editor closes.
19. Select the fields you wish to use for the record ID.

Note: Although records can contain both fields and structure, you can only select fields on the **Details** tab for the record ID.

20. Click Save on the editor tool bar to save the record.

To find other data format components that use the current data format record:

1. Select the **General** tab and click Where Used.
A list window containing **Data Formats** and **Loops** tabs opens.
2. Click the tab of each data format component to view a list of components containing the current data format record.

You can open any item in the list by double-clicking it.

An empty list window means that the data format record is not used in any other data format component of the list window's type.

Creating a data format structure

Use the data format structure editor to enter new structures into a data format dictionary or to edit existing structures. From the data format structure editor, you create and edit the structure's associations with fields and structures, as structures can contain fields and structures.

The data formats structure editor contains three tabs:

- The **General** tab contains fields for you to name the structure, select its dictionary, and set properties.
- The **Details** tab to contains fields for you add or change fields and structures associated with the selected structure and to edit the information about the association.
- The **Comments** tab to contains a field for you to type any comments.

The editor also contains a Where Used button, which lists other data format components that use the currently selected structure.

The following instructions are for creating a new structure. For information about viewing, copying, editing, renaming, deleting, and printing structures, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new structure when your application data has two or more contiguous fields that logically occur together. For example, a structure describing a date can contain fields for the year, month and day. In another example, Name, Address, City, State, and Zip occur three times in a single record in the same order. You can define those fields as a structure one time and say that it repeats three times in the record.

1. Select the **Structures** tab and click New on the tool bar.

Data format structure editor

The Data Format Structure editor opens with the **General** tab in front.

2. Type a name in the Structure Name field.

Notes:

- a. The name is displayed in uppercase letters.
 - b. You cannot type spaces within the name.
 - c. The name can be up to thirty characters long.
3. Select the data format dictionary in which you want the structure to occur using the Dictionary Name list.
 4. Enter a description of the structure in the Description field. This field is optional.
 5. Click the **Details** tab to enter information about the fields and structures contained in this structure.
 6. Click New. The Data Format Structure Detail editor opens with the **General** tab in front.
 7. Enter the position in the Position field. This field default is 1 and must be unique for each field or structure defined.
 8. Select the Type list.
 9. Either select the name of the field or structure from the list or type in a name to create a new field or structure. The list contains fields and structures that are in the same dictionary and use the same record ID information definition as this structure.

- If this is a field and you type in a new name for the field, click Insert and the Object Information section of the page becomes available. Go to step 10.
- If this is a structure and you type in a new name for the field, click Insert and the Occurrence Information section and the Description field of the Object Information section of the page become available. Go to step 13.

If you select an existing field or structure, the occurrence information or object information is filled in with that field or structures specific information. Go to step 16.

10. Enter a detailed description of the file in the Description field. This field is optional.
11. Select a data type from the Data Type list. See Table 9 on page 54 for more information about data types.
12. Specify the field length. The length of the field must be between 1 and 32767. Go to step 16.
13. Enter the number of time the structure repeats in the Occurs field.
14. Select the name of an existing Data Format Field within the Structure from the Occurs Depending On list. This field is used to indicate which value within the document determines how often a Structure represented by the row repeats.
15. Enter a detailed description of the file in the Description field. This field is optional. Go to step 16.
16. At this point you have two options:
 - To add the field or structure and continue adding records, click Insert and return to step 7.

- To add the field or structure and close the Data Format Structure Detail editor, click OK. The field or structure is added to the list window and the Data Format Structure Detail editor closes.

To find other data format components that use the current structure:

1. Select the **General** tab and click Where Used.
A list window containing **Records** and **Structures** tabs opens.
2. Click the tab of each data format component to view a list of components containing the current structure.

You can open any item in that component by double-clicking it.

An empty list window means that the structure is not used in any other data format component of the list window's type.

Creating a data format field

Use the data format field editor to enter new fields into a data format dictionary or to edit existing fields. From the data format field editor, you can set up and maintain properties of a data format field.

The data formats field editor contains two tabs:

- The **General** tab contains fields for you to enter the field name, select its dictionary, and set properties of the field.
- The **Comments** tab contains a field for you to type any comments.

The editor also contains a Where Used button, which lists other data format components that use the currently selected field.

The following instructions are for creating a new field. For information about viewing, copying, editing, renaming, deleting, and printing fields, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new field when your application data requires one.

1. Select the **Fields** tab and click New on the list window tool bar.
The Data Format Field editor opens with the **General** tab in front.
2. Type a name in the Field Name field. This is a required field, as indicated by the blue asterisk.

Notes:

- a. The name is displayed in uppercase letters.
 - b. You cannot type spaces within the name.
 - c. The name can be up to thirty characters long.
3. Select the data format dictionary in which you want the field to occur using the Dictionary Name list.
 4. Enter a description of the field in the Description field. This field is optional.
 5. Select the data type from the Data Type list. For definitions of WebSphere Data Interchange data types, see Table 9 on page 54.

Data format field editor

6. Type the length of the field in the Field Length field.
7. If this Field is for Data Formats that are part of a Send Map or a Receive Map, then you might need to complete the following fields:
 - a. Type the name of the literal you wish to use in the Literal field. For a list of WebSphere Data Interchange literals or mapping commands, search in WebSphere Data Interchange Client Help on the keyword "literals".
 - b. Literals and mapping commands are validated differently depending on whether they are used by Send maps or Receive maps. If you want this value validated, click either or both the Send Map Validation check box or the Receive Map Validation check box, depending which map types the literal or mapping command is used.
 - c. Enter a value into the Code List field to automatically add a Code List to a mapping in a Send Map or Receive Map when you map the Data Format Field. For information about creating User Code Lists, see "Creating a code list" on page 84.
8. When you have completed entering information, click Save on the tool bar to save the field.

After you have saved your field, the Field Name and Dictionary Name fields become read-only.

To find other data format components that use the current field:

1. Select the **General** tab and click Where Used.

A list window containing **Records** and **Structures** tabs opens.
2. Click the tab of each data format component to view a list of components containing the current field.

You can open any item in the list by double-clicking it.

An empty list window means that the field is not used in any other data format component of the list window's type.

Navigating between data format editors

WebSphere Data Interchange Client's data format editor are designed to provide maximum flexibility. You can move from editor to editor with ease so that you can tailor your navigation to the requirements of your work.

This section provides information about the various paths from one editor to the next.

Data format dictionary editor paths

The buttons at the bottom of the data format dictionary **General** tab generate list windows for each data format component that occurs in the current dictionary. From those list windows, you can open the editor for components in the list, where you can edit the current component or create a new one of that type.

Data Formats

The data formats associated with this dictionary.

Loops The loops associated with this dictionary.

Records

The records associated with this dictionary.

Structures

The structures associated with this dictionary.

Fields The fields associated with this dictionary.

Data format editor paths

From the data format editor, you can move to any other component of a data format. The most powerful navigational tool in the data format editor is its **Overview** tab, which displays a visual representation of your data format:

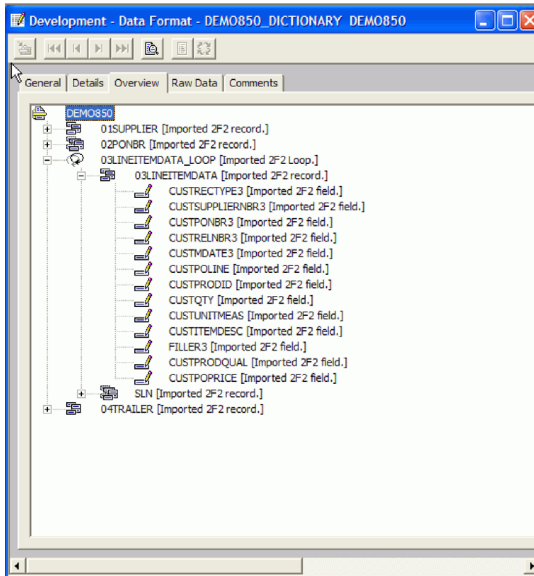






Table 8 defines the symbols on the data format editor Overview tab.

Table 8. Data format symbols

This symbol. . .	Represents:
	A loop
	A record
	A structure
	A field

Click the + sign to expand a section of the data format.

Click the - sign to collapse that section of the data format.

Navigating between editors

The **Expand** button expands the node which is currently selected. To expand the entire tree, select the root node and click **Expand**.

The **Overview** tab navigates through the data format as follows:

- Double-clicking a loop starts the data format loop editor for the loop on which you clicked.
- Double-clicking a record starts the data format record editor for the record on which you clicked.
- Double-clicking a structure starts the data format structure editor for the structure on which you clicked.
- Double-clicking a field starts the data format field editor for the field on which you clicked.

From the data format editor **Details** tab, you can start the data format loop editor and the data format record editor.

- To start the data format loop editor, double-click the number of the row containing the loop you wish to edit. You can also select the row and then click **Edit**.
- To start the data format record editor, double-click the number of the row containing the record you wish to edit. You can also select the row and then click **Edit**.

Data format loop editor paths

Within the data format loop editor, you can navigate to data formats, loops, and records, as follows:

From the data format loop editor **General** tab, you can view a list of data formats and loops in which the current loop is used.

Click **Where Used** to open a list window containing **Data Format** and **Data Format Loop** tabs. Double-clicking entries in those lists starts the respective editors.

From the data format loop editor **Details** tab, you can start the data format loop and data format record editors.

To start the data format loop editor, double-click the number of the row containing the loop you wish to edit. You can also select the row and then click **Edit**.

To start the data format record editor, double-click the number of the row containing the record you wish to edit. You can also select the row and then click **Edit**.

Data format record editor paths

Within the data format record editor, you can navigate to data formats, loops, structures, and fields, as follows:

From the data format record editor **General** tab, you can view a list of data formats and loops in which the current record is used.

Click **Where Used** to open a list window containing **Data Format** and **Data Format Loop** tabs. Double-clicking entries in those lists starts the respective editors.

From the data format record editor **Details** tab, you can start the data format structure and data format field editors.

To start the data format structure editor, double-click the number of the row containing the structure you wish to edit. You can also select the row and then click **Edit**.

To start the data format field editor, double-click the number of the row containing the field you wish to edit. You can also select the row and then click **Edit**.

Data format structure editor paths

Within the data format structure editor, you can navigate to records, structures, and fields, as follows:

From the data format structure editor **General** tab, you can view a list of records and structures in which the current structure is used.

Click **Where Used** to open a list window containing **Data Format Record** and **Data Format Structure** tabs. Double-clicking entries in those lists starts the respective editors.

From the data format structure editor **Details** tab, you can start the data format structure and data format field editors.

To start the data format structure editor, double-click the number of the row containing the structure you wish to edit. You can also select the row and then click **Edit**.

To start the data format field editor, double-click the number of the row containing the field you wish to edit. You can also select the row and then click **Edit**.

Data format field editor paths

Within the data format field editor, you can navigate to records and structures, as follows:

From the data format field editor **General** tab, you can view a list of records and structures in which the current field is used.

Click **Where Used** to open a list window containing **Data Format Record** and **Data Format Structure** tabs. Double-clicking entries in those lists starts the respective editors.

Reusing data format components

In the WebSphere Data Interchange Client, you can reuse data format components using the data format dictionary.

Reusing data format components

- When you create a component in a data format dictionary, you can use that component in any *other* data format associated with that dictionary, but you cannot reuse that component in the *same* component.
- You cannot use a loop or structure within itself, directly, or indirectly. That causes a circular reference.
- Names of all components within a dictionary must be unique.
- All data format names must be unique, regardless of dictionary.

Attention: Because you can reuse components, it is important that you check where each component is used when you change it. Your changes might propagate through several data formats. Use the Where Used button on the general tabs to see which data format components are affected by any change you make.

Understanding data types

The following table lists the valid data types permitted in data format fields. These data types describe the contents of the fields in your application's data. It also shows the valid mapping data type associated with the data format data type. The EDI standard data types column shows equivalent EDI standard data types.

Most data format data types use the same format for storing, displaying, and printing data. However, sometimes the storage format is different from the format used to display or print data. These differences are noted in Table 9.

Note: Data types that contain binary data are not used for comma-separated data formats. This can result in unpredictable behavior because the binary data can be interpreted as record or field delimiters.

Table 9. Data types for data formats

Data format	EDI standard data types	Data Format description
A	A	Alphabetic
	AN	Any combination of characters from the ALHPANUM table, except the digits 0–9.
	ID	
AC	A	Application control
	AN	A field that contains a control number by which the application identifies the transaction. A purchase order number is an example. The data itself is alphanumeric. A data format can contain only one field of this type.
	ID	
	Nn	During transaction mapping, you can specify the application control as a concatenation of up to eight fields. The concatenated application control overrides the AC data type.
	Rn	
	DT	This data type does not apply to record IDs.
TM	An AC data type is assumed to be the same as AN during the value validation at translate time.	

Table 9. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
AN	A AN ID Nn Rn DT TM	Alphanumeric You can use any combination of characters up to the length of the field.
Bn	AN ID Nn Rn DT TM	Binary (unsigned) Storage format: Data with a binary format with n implied decimal places. A value of 2.3 defined as a 2-byte B2 field would be stored as 1110 0110 (X'E6' or decimal 230). This is the same format as IT or In data, but binary data is not signed and, therefore, all values are considered positive.
BN	AN ID Nn Rn DT TM	Binary (unsigned) Any combination of 0–9 without a sign (+ or -). Storage format: The binary equivalent of a numeric value in either two or four bytes, depending on the length of the field. Example: The value 23 is stored as 0000 0000 0001 0111 (X'0017').
CH	A AN ID Nn Rn DT TM	Character Any combination of characters up to the length of the field.

Data types for data formats

Table 9. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
DT	DT	<p>Date; does not apply to record IDs</p> <p>A string of 5 to 8 digits, depending on the date format that is used. The acceptable date formats are:</p> <p style="padding-left: 40px;">ddmmyy, ddmyyyy, ddyymm, ddyyyyym, ddy, ddyyy</p> <p style="padding-left: 40px;">mmddy, mmddyyy, mmyy, mmyyyd, mmyyyd</p> <p style="padding-left: 40px;">yyymm, yyym, yydmm, yyydmm, yyd, yyyd</p>
FN		<p>File name.</p> <p>A field with data type FN is treated as if the data type were AN.</p> <p>In send and Receive maps, specifies a field that contains the name of a file whose entire contents are mapped to a binary segment.</p>
Hn	AN ID Nn Rn DT TM	<p>Hexadecimal</p> <p>Hexadecimal data with n implied decimal places.</p> <p>This format is treated as a Bn field when mapped to a numeric data element and as an HX field when mapped to an alpha data element.</p>
HX	AN ID Nn Rn DT TM	<p>Hexadecimal</p> <p>Any combination of 0–9 and A–F up to twice the length of the field.</p> <p>Storage format: Hexadecimal, where the length of the field determines the number of bytes used to hold the value.</p>
ID		<p>Identifier</p> <p>The ID data type is equivalent to an AN data type.</p>

Table 9. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
In	AN ID Nn Rn DT TM	Integer (signed) Storage format: Data with a binary format with n implied decimal places. A value of 2.3 defined as a 4-byte I2 field would be stored as 0000 0000 1110 0110 (X'E6' or decimal 230).
IT	AN ID Nn Rn DT TM	Integer (signed) Storage format: The binary equivalent for a positive number or the two's complement binary equivalent for a negative number, in two or four bytes, depending on the length of the field. Example: The value +23 is stored as 0000 0000 0001 0111 (X'0017'). The value -23 is stored as 1111 1111 1110 1001 (X'FFE9').
IV		Incrementing Value An EDI Standard Data Element, such as a message reference number, that starts at 1 and increases by 1 for each occurrence. This data type does not apply to record identifiers. Note: This format is used in send and Receive maps only.
Ln	AN ID Nn Rn DT TM	Decimal (leading sign) Zoned decimal data with n implied decimal places and a leading sign.

Data types for data formats

Table 9. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
N	AN	Numeric
	ID	Any combination of 0–9 and an optional sign (+ or –). The length includes the sign.
	Nn	When mapping data elements defined as data type N in UN/EDIFACT standards, use data type R.
	Rn	
	DT	
	TM	
Nn	AN	Numeric
	ID	Any combination of 0–9, an implied decimal point with n places to the right of the decimal, and an optional sign (+ or –). Using N alone is the same as using NO (N zero). The length includes the sign.
	Nn	Example: N2 for a value of 23949 is interpreted as 239.49.
	Rn	
	DT	
	TM	
PD	AN	Packed decimal
	ID	Any combination of 0–9 with a sign (+ or –). The length defines the number of bytes used to hold the value in external format (minus the sign position).
	Nn	Storage format: The packed decimal equivalent, followed by the sign in the low-order 4 bits of the last byte. The sign is either 1111, 1100, or 1010 for a positive value; or, 1101 or 1011 for a negative value.
	Rn	
	DT	
	TM	
Pn	AN	Packed decimal
	ID	Packed decimal data with n implied decimal places.
	Nn	
	Rn	
	DT	
	TM	

Table 9. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
PW		<p>Password</p> <p>A password used in the interchange or functional group header. This data type does not apply to record identifiers.</p> <p>This data type does not apply to record identifiers.</p>
R	AN ID Nn Rn	<p>Real</p> <p>Numeric data that requires a decimal point for fractional values. The decimal point is optional for integers. A sign (+ or –) is optional for positive numbers. Positive is assumed if a sign is not present. The length includes the decimal point and sign if they are present.</p> <p>Scientific notation with exponent and mantissa formatting is used.</p> <p>Use this data when mapping data elements defined as data type N in UN/EDIFACT standards.</p> <p>Examples: 23.949, +23.949, –23949, –39846.7, 50E+4.</p>
Rn	AN ID Nn Rn	<p>Real</p> <p>Signed or unsigned numeric data with a minimum for n significant decimal places. The length includes the decimal point and sign. Any combination of 0–9 with a sign (+ or –).</p>
TM		<p>Time</p> <p>A string of four digits in the form hhmm or six digits in the form hhmmss, expressed in the 24-hour clock format, where the hour is specified as 00 to 23 for X12 and 00 to 24 for EDIFACT.</p> <p>This data type does not apply to record IDs.</p>
ZD	AN ID Nn Rn DT TM	<p>Zoned decimal</p> <p>Any combination of 0–9 with a sign (+ or –). The length defines the number of characters used to represent the value in the external format. The external length requires an extra position for the sign.</p> <p>Storage format: The zoned decimal equivalent in the low-order 4 bits of a byte and 1111 in the high-order 4 bits. The sign displays in the high-order 4 bits of the low-order byte and is either 1100 for a positive value or 1101 for a negative value. The length of the field determines the number of bytes used to store the value.</p> <p>Example: The value +123 is stored as 1111 0001 1111 0010 1100 0011 (X'F1F2F3'). The value –123 is stored as 1111 0001 1111 0010 1101 0011 (X'F1F2D3').</p>

Data types for data formats

Table 9. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
Zn	AN ID Nn Rn DT TM	Zoned decimal Zoned decimal data with n implied decimal places and a trailing sign. Any combination of 0–9 with a sign (+ or –).

Chapter 5. Extensible Markup Language

Extensible Markup Language (XML) is a popular way to represent structured documents and data. XML defines the syntax used to represent the data, including information such as how to tell an element name from an element value. However, it does not define the semantics or structure of the data. For example, it does not specify where a purchase order number appears, or even whether it is part of any particular document.

Note: XML is handled natively in WebSphere Data Interchange Version 3.3. It is not necessary to use the DTD Conversion Utility with WebSphere Data Interchange Version 3.2 and later unless you are updating a DTD that was converted in Version 3.1. When using XML in WebSphere Data Interchange Version 3.2 and later, the DTD is imported directly into the WebSphere Data Interchange Client.

The structure of an XML document is defined by a Document Type Definition (DTD) or schema. The syntax of the DTD is defined as part of the XML language or schema. The DTD or schema provides a list of all components included in the XML document and their relationship to each other. For example, a DTD or schema can state that the Header element of the document contains a PONum element. The meaning of the PONum element can be described either in the comments within the DTD or schema, a separate document, or both.

Unlike EDI standards where there are a small number of dominant EDI standard formats that define the document structure, there are numerous different XML DTDs and schemas. Also, because XML is extensible, users are free to create their own DTDs and schemas if they choose. Instead of restricting users to a fixed subset of DTDs and schemas, WebSphere Data Interchange imports the DTDs and schemas you use. You can obtain your DTDs and schemas from various sources, such as industry groups, standards bodies, vendors, trading partners, or you can create them yourself. When the DTDs and schemas are imported into WebSphere Data Interchange, you can map the documents described by those DTDs and schemas.

Note: Imported DTDs and schemas are used only for Data Transformation maps. They cannot be used for send and Receive maps.

WebSphere Data Interchange uses XML Dictionaries as way to logically group a set of related DTDs and schemas. A DTD or schema name must be unique within its XML dictionary, but does not need be unique across all XML dictionaries. A DTD or schema can use external references to refer to other DTDs and schemas within the same dictionary. A dictionary cannot contain both a schema and a DTD with the same name. Use the XML dictionary editor to create and maintain an XML dictionary.

The XML list window provides access to the XML dictionaries, DTDs, schemas, and namespaces. The window opens when you click XML on the navigator bar contains four tabs, as follows:

- The **XML Dictionary** tab provides access to the XML dictionary list window and XML dictionary editor.
- The **DTDs** tab provides access to the DTDs list window and the DTDs editor.

- The **Schemas** tab provides access to the schemas list window and the schema editor.
- The **Namespaces** tab provides access to the namespaces list window and the namespaces editor.

Accessing XML editor

You access the component editors in essentially the same way. The editors display on tabs in the XML list window, as follows.

Note: You cannot change the DTD or schema itself, only the WebSphere Data Interchange properties associated with the DTD or schema. To change a DTD or schema, edit the original DTD or schema outside of WebSphere Data Interchange using a text editor or a DTD or schema editor, then re-import the DTD or schema into WebSphere Data Interchange. For send and Receive maps, see Appendix D, “DTD Conversion Utility,” on page 321.

To access an XML editor:

1. Click XML on the WebSphere Data Interchange Navigator bar.
The XML list window, which contains tabs for the components, is displayed.
2. Click the tab of the XML component you wish to work with.
The list window for that component is displayed.
This window opens a list of existing XML components. A list of XML dictionaries is shown. Each row contains information about a component; each column contains data stored in that component. Information in the columns displays in fields in the editor.
3. To view an item or to add or change information in an item, double-click the row of the item you wish to work with.
The editor is displayed. You add information or make changes to the component through its tabs, as described in the following sections.

The following sections contain detailed procedures for creating XML dictionaries and importing DTDs and schemas. For information about viewing, copying, editing, renaming, deleting, exporting, and printing components, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Creating the XML dictionary

An XML dictionary is a named group of related DTDs. The relationship between the DTDs in a dictionary is not fixed. For example, you can group DTDs that refer to one another or DTDs created and maintained by one organization. You can put all DTDs from a particular XML format in one dictionary.

Create a new dictionary when you want to create a logical group of DTDs.

1. Select the **XML Dictionary** tab and click New on the tool bar. The XML dictionary editor is displayed with the **General** tab in front.
2. Type a name in the Dictionary Name field.

Notes:

- a. The name is displayed in uppercase letters.
- b. You cannot type spaces within the name.
- c. The name can be up to 30 characters long.
3. Enter a description of the XML dictionary in the Description field. This field is optional.
4. Click Save on the tool bar to save the dictionary.

Importing and defining a DTD or schema file

After you have created an XML dictionary, DTDs and schemas can be imported into WebSphere Data Interchange. Import a new DTD object when you plan to process an XML document defined by an XML DTD which has not previously been imported into WebSphere Data Interchange

Note: DTD objects are assigned to an XML Dictionary. The XML Dictionary must be created before importing the DTD into WebSphere Data Interchange.

To import a new DTD or schema object:

1. Select either the **XML** or **Schemas** tab.
2. Select File > Open Import File from the File menu. This opens the Select Import File dialog.
3. In the Files of Type list, select either XML DTD File or XML Schema File. The window changes so only files with the appropriate type display.
4. Use the dialog to select a directory and file name of the DTD or Schema to be imported. Click Open after you select your DTD or Schema file. The Select a System dialog opens if you have more than one System defined to WebSphere Data Interchange Client. If you only have one System defined, the Import XML DTD dialog opens.
5. If the Select a System dialog opens, select the WebSphere Data Interchange system you wish to import the DTD or schema into, then click OK. An import dialog opens.
6. On the import dialog, enter the appropriate information for the DTD or Schema and click Import.

At this point the object is imported into WebSphere Data Interchange. A DTD or Schema object name must be unique within its XML Dictionary, but does not need be unique across all XML Dictionaries. An imported DTD replaces any existing DTD object or Schema object that has the same name. DTDs or Schemas that represent a document commonly need to have elements that identify the sending and receiving trading partners. Use the DTD editor to identify the elements that contain the information identifying the sending and receiving trading partners.

WebSphere Data Interchange needs to know the name of the Trading Partner profile representing the sender and receiver of the document. Certain elements in an XML document are used to determine who sent a document and who the receiver of the document is. These elements are used to determine the name of a Trading Partner

Importing and defining a DTD file

profiles representing the sending and receiving trading partners. The elements are typically named things like <Partner> or <Vendor> In addition, names can be two parts, like people names. The first part is called the qualifier and usually denotes the name space to be used to interpret the second part of the name, usually referred to as the ID. An example of a name space for companies would be DUNS numbers.

WebSphere Data Interchange does not know what elements (if any) in an arbitrary XML document denote the sender and receiver of the document. You must identify the elements to WebSphere Data Interchange so it can determine the sender's Trading Partner profile and receiver's Trading Partner profile. Use the Sender Qualifier Element and Sender ID Element fields to specify which elements to use to help determine the sender's Trading Partner profile. Use the Receiver Qualifier Element and Receiver ID Element fields to specify which elements to use to help determine the receiver's Trading Partner profile. The Internal Trading Partner ID field can also be used to help determine the receiver's Trading Partner profile in a source XML document. The Sender Translation Table field can be used with the Sender Qualifier and ID Element fields to help determine the name of the Trading Partner profile representing the sender. The Receiver Translation Table field can be used with the Receiver Qualifier and ID Element fields to help determine the name of the Trading Partner profile representing the receiver.

The DTD and schema editors are used to maintain properties of the DTD or schema and to view the DTD or schema. It cannot be used to change the DTD or schema itself. Change the original DTD or schema using a text editor or a DTD editor.

To define information such as a description or root element for a DTD or schema:

1. Click XML on the WebSphere Data Interchange Client Navigator bar. The XML list window is opens.
2. Click the DTDs or Schemas tab.
The list window for DTDs or schemas opens.
3. Double-click the row of the DTD or schema you wish to work with.
The **General** tab in the DTD or schema editor opens.
4. Complete the fields on the **General** tab as follows:
 - a. The DTD Name and XML Dictionary are used to uniquely identify the DTD object. These fields cannot be changed on existing DTD objects, except by using WebSphere Data Interchange's rename feature. Use these fields to refer to this object in maps, and WebSphere Data Interchange PERFORM commands.
 - b. Enter a description of the object. This field is optional.
 - c. Enter the Root Element if the object is to be used in maps.
 - d. If there are elements within the object that contain the sender qualifier and sender ID, the element paths need to be entered for those elements in the appropriate fields. It is recommended that you set these values using the Overview tab.
 - e. If there is an element within the DTD that contains the internal trading partner ID, the element's path needs to be entered into the appropriate field. It is recommended that you set this value using the Overview tab.

- f. If there are elements within the DTD that contain the receiver qualifier and receiver ID, the element paths need to be entered for those elements in the appropriate fields. It is recommended that you set these values using the Overview tab.
 - g. If the XML document split feature is going to be used with the documents defined by this DTD, then enter the name of the elements that identify the header section, messages, and trailer section of the documents.
 - h. Click Save on the toolbar to save the object.
5. Click the **View** tab on the DTD or schema View Window.
The **View** opens, showing the contents of the DTD or schema.
 6. Click the **Overview** tab on the DTD or schema View Window.
The **Overview** opens, showing a graphical view of the DTD or schema.

Creating an XML Namespace

The Namespace object in WebSphere Data Interchange is used to identify an XML namespace to WebSphere Data Interchange. Namespaces are used primarily within XML schemas and are generally created automatically when XML schemas are imported into WebSphere Data Interchange Client. Namespace objects are located in the XML Functional Area. The Namespaces list window is used to list and perform maintenance functions on WebSphere Data Interchange Namespace objects. A Namespace is named after the Uniform Resource Identifier (URI) it represents. These names typically look like the familiar URLs on the World Wide Web. Each Namespace object is contained within an XML Dictionary.

Note: A Namespace object name must be unique within its XML Dictionary, but does not need to be unique across all XML Dictionaries.

WebSphere Data Interchange Client scans a schema looking for the namespace identifier, xmlns, while the schema is being imported. When the identifier is encountered, WebSphere Data Interchange determines if a corresponding Namespace object exists within the same XML Dictionary. If one does not exist, a corresponding Namespace object is created within the XML Dictionary. The prefix and other information associated with Namespace objects can be changed later.

You can create a Namespace object if needed. To create a new Namespace object:

1. Select the **Namespaces** tab and click New on the list window toolbar. The Namespace editor opens with the **General** tab in front.
2. Type in the URI for the Namespace object. It serves as the name of the Namespace object.
3. Use the Dictionary Name list to select the XML Dictionary for the Namespace object.
4. Enter a description, prefix, and schema location as needed. You enter the prefix or schema location, or both. These fields are optional.
5. Select the **Comments** tab and add any comments you have about the Namespace object.

Creating a namespace

6. Click Save on the editor toolbar. WebSphere Data Interchange Client saves the new Namespace object to the System.
7. Close the editor when you are done.

XML Document processing

The default XML document processing using Data Transformation enables the mapping of a single XML document to a single target document. Many XML source documents resemble EDI data. A single XML document contains header type information, multiple messages, and trailer type information. It is desirable, for example, to map the XML source document to multiple EDI target documents. To achieve this, using the default XML processing, a double transformation process is needed to transform the XML document to an intermediate document, for example data format, with a second transformation process to map the intermediate document to the EDI document.

Options exist to split a single XML document based on a defined compound XML element and reconstructed before the document enters the Data Transformation message flow.

WebSphere Data Interchange XML DTD and Schema definitions contain an **Overview** tab to display a visual layout of the DTD or schema. You can right click elements and use functions on the popup menu to set certain fields in the **General** tab. A right click a simple element opens a popup menu to set elements that contain the sender and receiver ID and qualifier paths. A right click a compound elements opens a popup menu to set XML document split element IDs.

Note: The split element identification is not a path, it is an element ID.

There are three elements that can be defined to split the XML document.

- Element identifying the header area in the XML document
- Element identifying the individual messages (split area)
- Element identifying the trailer area in the XML document.

These definitions are used to split and reconstruct the XML documents before they are placed in the Data Transformation message flow. The element identifying the individual messages is required to split the source XML document. If the element identification is not defined, the source XML document is not be split. If the header area is not defined, the beginning of the XML document up to the element identifying the message is used to construct a header area for the split document. If the trailer area is not defined, the end root element are used as the trailer area for the split document. If the trailer area is defined and is actually a terminating element in the XML source input, then the right click to define this using WebSphere Data Interchange is to right click the compound element (that begins the trailer element), define the element as the trailer element, and check the Element Terminator Indicates Start of Trailer Section box on the **General** tab.

An XML source document property MsgSplitCnt is available and can be used to identify the number of documents split within each header/message/trailer split. The MsgSplitCnt property is set to zero until the last split message is processed.

MsgSplitCnt property is reset with each new trailer/header identification. The MsgSplitCnt property can be used, for example, to MapChain() to a summary mapping by counting the number of source messages processed using a global variable within the source document mapping and comparing this to the MsgSplitCnt property.

The InputMsgCnt source document property identifies the number of input messages processed and is available in Data Transformation mapping.

During processing, the XML source input message is read and stored in a buffer. The root element is identified (from the input message) and the WebSphere Data Interchange DTD or Schema definition is retrieved to determine if the source XML document has been defined as a split document. This is done with all XML input source messages and can be removed by specifying the PERFORM keyword XMLSPLIT(N).

During the XML document definition retrieval, if there are multiple DTD or Schema definitions defining the same root element, it might be necessary to use PERFORM keywords DICTIONARY or DOCUMENT to identify the specific DTD or Schema being used for processing.

If the XML document is to be split, the header area is identified and stored in a header area buffer, the trailer area is identified and stored in a trailer area buffer. The header area is the beginning of the XML document and the Header element identification up to the first Message element identification. The trailer area is defined as the Trailer element identification up to the next header element identification. The area between the first message element identification up to the trailer element identification are written out to the XMLWORK file. During reconstruction, each split message is read from the XMLWORK file. The header/message/trailer are constructed and sent through the Data Transformation message flow as individual documents.

Note: The XMLWORK file must be allocated.

Example 1

The sample XML document contains information about three companies. The expanded elements (those preceded by a dash) show the first company element contains information about the company and information about four employees of the company. If each employee element needs to be translated into its own document, then the employee element would be listed as the Message Element on the **General** tab of the Schema editor. The Header Element would be the company element. The Trailer Element would be the employee-list element with the Element Terminator Indicates Start of Trailer Section check box set.

Creating a namespace

```
<?xml version="1.0" encoding="UTF-8" ?>
- <root-element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ThisDoc.xsd">
  - <company>                                     <=== Header Element
    - <company-details>
      + <company-name>
      + <company-address>
    </company-details>
    - <employee-list>
      + <employee>                               <==== Message Element (Split here)
      + <employee>
      + <employee>
      + <employee>
    </employee-list>                             <=== Trailer Element
  </company>                                     (Note: end of header area)
+ <company>
+ <company>
</root-element>
```

In the result would be one document like the following for each employee contained in the source document.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <root-element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ThisDoc.xsd">
  - <company>
    - <company-details>
      <company-name>
      + <company-address>
    </company-details>
    - <employee-list>
      + <employee>
    </employee-list>
  </company>
</root-element>
```

Example 2

The sample XML document contains information about three companies. The expanded elements (those preceded by a dash) show the first company element contains information about the company and information about four employees of the company. If each employee element needs to be translated into its own document, then the employee element would be listed as the Message Element on the **General** tab of the Schema editor. The Header Element would be the company element. The Trailer Element would be the employee-list element with the Element Terminator Indicates Start of Trailer Section check box set.

```

<?xml version="1.0" encoding="UTF-8" ?>
- <root-element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ThisDoc.xsd">
  - <company>          <=== Header Element
    - <company-details>
      + <company-name>
      + <company-address>
    </company-details>
  </company>          (Note: end of header area)
- <employee-list>
  + <employee>       <==== Message Element (Split here)
  + <employee>
  + <employee>
  + <employee>
  </employee-list>   <=== Trailer Element
+ <company>
+ <company>
</root-element>

```

In the result would be one document like the following for each employee contained in the source document.

```

<?xml version="1.0" encoding="UTF-8" ?>
- <root-element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ThisDoc.xsd">
  - <company>
    - <company-details>
      <company-name>
      + <company-address>
    </company-details>
  </company>
- <employee-list>
  + <employee>
  </employee-list>
</root-element>

```

Example 3

The sample XML document contains information about three companies. The expanded elements (those preceded by a dash) show the first company element contains information about the company and information about four employees of the company. If each employee element needs to be translated into its own document, then the employee element would be listed as the Message Element on the **General** tab of the Schema editor. The Header Element would be the company element.

Note: With no trailer area, only the first header element occurrence can be identified. All employee elements are split under the first company occurrence. `MsgSplitCnt` property is a total and set with the last employee split processed.

Creating a namespace

```
<?xml version="1.0" encoding="UTF-8" ?>
- <root-element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ThisDoc.xsd">
  - <company>                                     <=== Header Element
    - <company-details>
      + <company-name>
      + <company-address>
    </company-details>
  </company>                                     (Note: end of header area)
+ <employee>                                     <==== Message Element (Split here)
+ <employee>
+ <employee>
+ <employee>
+ <company>
+ <company>
</root-element>
```

In the result would be one document like the following for each employee contained in the source document.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <root-element xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ThisDoc.xsd">
  - <company>
    - <company-details>
      <company-name>
      + <company-address>
    </company-details>
  </company>
+ <employee></root-element>
```

Chapter 6. EDI standards

EDI standards provide a common document layout that trading partners use to exchange data between their computer applications. In essence, EDI standards provide the building blocks for electronic versions of common business documents.

WebSphere Data Interchange translates data from one document layout to another. Commonly, the source document layout describes a document from a business application, which is translated into an EDI standard transaction for transmission to a trading partner. Conversely, WebSphere Data Interchange commonly translates data received in an EDI standard transaction into a format used by a business application. When you install WebSphere Data Interchange, you receive copies of EDI standards currently approved by the primary EDI standards organizations. You can also download EDI standards from the WebSphere Data Interchange Web site:

<http://www.ibm.com/websphere/datainterchange>

For instructions on how to install EDI standards, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

To begin exchanging documents with a trading partner using an EDI standard transaction, you must select an EDI standard transaction that corresponds to the information you want to send or receive. Ideally, you and your trading partner can agree on an EDI standard transaction that requires no customization to meet your needs, but this is not always possible. Imagine, for example, that you and your trading partner need to exchange specific information that is not included in existing EDI standards. With WebSphere Data Interchange Client, you can customize currently approved EDI standards so that they fit your needs.

Attention: When altering EDI standards, work in close partnership with your trading partners. If you customize EDI standards without informing your trading partners of the changes, they might not be able to process the transactions you send.

Terminology

The terms transaction set in ANSI ASC X12 and message in UN/EDIFACT are equivalent to transaction in WebSphere Data Interchange.

An EDI standard structures data into two basic categories: envelopes and transactions.

Envelopes

Envelopes are made up of the control structures that wrap data for communications to trading partners. UN/EDIFACT refers to envelope standards as service segments, and ASC X12 refers to them as envelope interchange control segments. Envelope standards also specify the default delimiters used in the EDI standard data, such as the data element delimiter, subelement delimiter, and segment delimiter. See “Editing an envelope standard” on page 84

Note: Envelopes are used for send and Receive maps only.

Transactions

Transactions correspond to business documents such as purchase orders or invoices. An EDI standard contains one definition for each unique segment and data element that occur in a transaction. These segments and data elements are then used in as many transaction sets as necessary.

A transaction consists of the following components:

- EDI standard dictionary

An EDI standard dictionary contains information about all of the transaction sets, segments, and data elements that comprise the specific version and release of an EDI standard. For detailed information about a particular EDI standard, consult the appropriate EDI standards manuals.

When you install WebSphere Data Interchange, you receive a copy of EDI standard dictionaries currently approved by the primary EDI standards organizations. For information about how to create dictionaries, see “Creating the EDI standard dictionary” on page 74.

- Transaction set

Transaction sets represent business documents such as invoices or purchase orders. Transaction sets are called messages in UN/EDIFACT and transactions in WebSphere Data Interchange. A transaction set contains segments. These segments are sometimes grouped into tables representing heading information (such as billing address), detail information (such as line items from a purchase order), and trailing information (such as totals). For more information about how to create or edit transactions, see “Creating a transaction” on page 76.

- Segment

A transaction set is composed of segments. In essence, each line of a business document corresponds to a segment in the EDI transaction set. Segments begin with a segment identifier assigned by the EDI standard. They are either mandatory, conditional, optional, or floating (floating segments can display anywhere in the transaction). All segments except for floating segments display in a fixed sequence for a given transaction.

Segments can repeat within a transaction up to the number of times specified by the EDI standard. Groups of segments, such as the group which makes up a name and address, can form a loop. Loops are identified by a loop ID. Entire loops can be repeated in succession up to the number of times specified by the EDI standard. For information about how to create or edit segments, see “Creating a segment” on page 79.

- Data element

A segment is composed of data elements and composite data elements, which represent the individual units of data found in business documents, such as quantity ordered or unit price. Data elements display in a sequence specified by the EDI standard and are separated by a delimiting character, such as an asterisk. They have a minimum and maximum length, and are either mandatory, conditional, or optional.

WebSphere Data Interchange Client also supports composite data elements. Composite data elements are composed of a group of logically related simple data elements. A Composite Unit of Measure, for example, is a combination of Unit of Basis for Measurement, Component, and Multiplier. Composite data elements are defined in the EDI standards.

All EDI standard data elements must be of a data type prescribed by the EDI standard, such as date, time, and alphanumeric. Identifiers, such as data type ID, must contain one of the codes prescribed by the EDI standard. The EDI standard specifies the list of acceptable codes, which you can customize. For information about how to create or edit data elements, see “Creating a data element” on page 81.

- Code lists

A code list is a list of acceptable values for segments or data elements that can only contain certain values. If you include a segment or data element that can only contain certain values in the transaction set you are creating, enter all acceptable values into a code list.

When the WebSphere Data Interchange Server validates or translates a document and the use of a Code List is specified to validate a piece of data, the specified Code List is searched to see if it contains the value. If the field contains a value that does not display in the code list, WebSphere Data Interchange returns a processing error. For information about how to create code lists, see “Creating a code list” on page 84.

Terminology

Code lists are called Validation Tables on the WebSphere Data Interchange Server.

Using the EDI standard editors

Use the EDI standard editors to create or maintain the various components that make up an EDI standard. Although you can use the WebSphere Data Interchange Client standard editors to create a completely new EDI standard, most users are not likely to do that.

The EDI standard editors are most often used to modify existing EDI standards to meet a company’s needs. This section provides a generic description of the procedures for using the EDI standards editors.

Use the EDI standard list window to gain access to the EDI standards component editors. Each component editor corresponds to a tab on the EDI standard list window, as follows:

- The **EDI Standard Dictionary** tab provides access to the EDI standard dictionary list window and dictionary editor.
- The **Transactions** tab provides access to the transactions list window and the EDI standard transaction editor.
- The **Segments** tab provides access to the segments list window and EDI standard segment editor.

EDI standard editors

- The **Data Elements** tab provides access to the data elements list window and EDI standard data element editor.
- The **Code Lists** tab provides access to the code lists list window and code list editor.
- The **Envelope Standards** tab provides the ability to view or change an existing Envelope Standard.
- The **Envelope Control Strings** tab displays a list of Envelope Control Strings selected by the current Query

To accessing the EDI Standards editor:

1. Click EDI Standards on the WebSphere Data Interchange Client Navigator bar.

The EDI Standards list window is opens.

2. Click the tab of the EDI standard component you wish to work with.

The list window for that component opens.

This window opens a list of existing items in an EDI standards component. Each row contains information about an item; each column contains data stored in that item. Information in the columns displays in fields in the editor. The list window also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the page to scroll to the right or left. To alter the columns that display on the page, or to change which query is executed to produce the list, click Modify Window Properties. To create new queries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01

3. To view an item or to add or change information in an item, double-click the row of the item.

The editor opens. You add information or make changes to the EDI standard item using its editor, as described in the following sections.

The following instructions are for creating EDI standards components. For information about viewing, copying, editing, renaming, deleting, and printing components, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. For information about exporting components, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. (EDI Standard dictionaries, EDI standard transactions, and code lists are the only EDI standards components that can be exported.)

The EDI standard component editors are described in the following sections in the order in which you use them when creating an EDI standard.

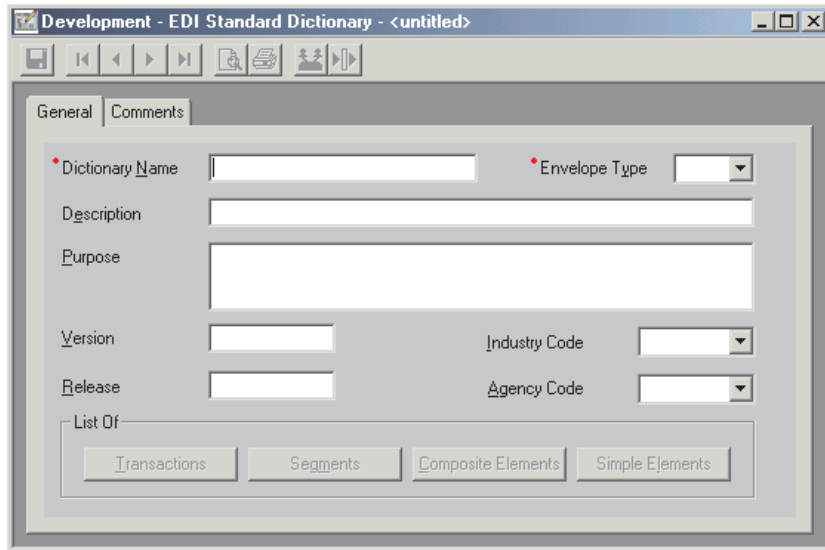
Creating the EDI standard dictionary

An EDI standard dictionary is a named group of other related components.

Detailed procedures for creating a new dictionary follow. For information about viewing, copying, editing, renaming, deleting, and printing dictionaries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. For information about exporting dictionaries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new dictionary when you want to create your own customized EDI standard.

1. Select the **EDI Standard Dictionary** tab and click New on the tool bar.
The EDI standard dictionary editor opens with the **General** tab in front.



2. Type a name in the Dictionary Name field.

Notes:

- a. The name is displayed in uppercase letters.
 - b. You cannot type spaces within the name.
 - c. The name can be 8 characters in length.
3. Enter a description of the EDI standard dictionary in the Description field. This field is optional.
 4. Select an envelope type from the Envelope Type list.

Note: When you save your dictionary, the Transactions, Segments, Composite Elements, and Simple Elements buttons in the List Of box become available. Click those buttons to display list windows that contain the components associated with this dictionary. When you first create a dictionary, the lists are empty.

5. Enter the version and release of the dictionary. These fields are optional.
6. Select an Industry Code from the list provided; if a list is not available, you can type a name in the list. The available codes are:

RAIL Association of American Railroads

UCS Uniform Communication Standard

VICS Voluntary Inter-Industry Communications Standard

7. Select an Agency Code from the list provided; if a list is not available, you can type a name in the list. The available codes are:

EDI standard dictionary

T	TDCC, ODETTE
UN	UN/EDIFACT
X	ASC X12, RAIL, UCS, VICS

8. Click Save on the tool bar to save the dictionary.

To view lists of EDI standard components in the current dictionary:

1. Click the button in the List Of group box corresponding to the component you wish to view.

A list window displaying EDI standard components associated with this dictionary opens.

- The Transactions button opens the EDI Standard Transactions list window.
- The Segments button opens the EDI Standard Segments list window.
- The Composite Elements button opens the EDI Standard Data Elements list window showing only composite data elements.
- The Simple Elements button opens the EDI Standard Data Elements list window showing only simple data elements (data elements that are not composite data elements).

2. You can open any item in the list by double-clicking on it.

Creating a transaction

The EDI standard transaction editor defines and structures the components that make up a transaction. The EDI standard transaction editor contains four tabs:

- The **General** tab contains the fields for you to enter and change transaction properties, such as name and description.
- The **Details** tab contains the fields for you to add or change the usage of segments associated with the selected transaction.
- The **Notes** tab displays a list window of the notes associated with specific transaction. You can also add a new note from this list window.
- The **Comments** tab contains a field for you to type any comments.

The following instructions are for creating a new transaction. For information about viewing, copying, editing, renaming, deleting, and printing transactions, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new transaction when the transactions shipped as part of the EDI standards do not meet your business needs. This editor also modifies existing transactions.

1. Selects the **Transactions** tab and click New on the tool bar.
The EDI Standard Transaction editor opens with the **General** tab in front.
2. Type a name in the Transaction field.

Notes:

- a. The name is displayed in numbers and capital letters.
- b. You cannot type spaces within the name.

- c. The name can be up to 8 characters long and can contain alphanumeric characters and any of the special characters “!@#%&*()_-=+':;<,.>?”.
3. Select the dictionary in which you want the transaction to occur through the Dictionary list.
4. Enter a description of the transaction in the Description field and a brief summary of the transaction’s purpose in the Purpose field. These fields are optional.
5. Enter a Functional Group, such as IN for invoice.
6. Select the **Details** tab and click New. The Add EDI Standard Transaction Detail editor opens.
7. Complete the optional fields you need. The following table describes the fields and their values.

Table 10. Add EDI Standard Transaction Detail editor fields

Field	Description
Table	The value in this field indicates the table in which the Segment belongs. A table is a section of the transaction. Many Transactions have only a single table (numbered 1. Some Transactions have a grouping that uses the value 1 to represent header information in the document, the value 2 to represent the details of the document, and the value 3 to represent the trailer information of the document. The valid values for the field are 1 to 32,767.
Position	The value in this field indicates the relative position of the Segment within the specific table of the Transaction. Each value must be unique within the table. The values do not need be sequential. When the Transaction is saved the list of Segments are sorted based on the table and position number. The valid values for this field are 1 to 32,000.
Segment	This column is used to identify the name of the Segment. Select a name from the list of available Segments.
Requirement Designator	Use this column to indicate whether the Segment is optional, mandatory, conditional, or floating. The conditional value indicates that the Segment might present depending on semantic conditions. Use the EDI Standard Notes action to see a list of Notes or to maintain Notes for the Transaction.
Maximum Repeat	This value indicates the maximum number of times the Segment can occur at this position. The value can be from 1 to 9,998. This field is disabled if the Unlimited Repeat is selected. The first Segment within a Loop must have the Maximum Repeat column set to 1.
Unlimited Repeat	This check box is used to indicate that the Segment can repeat infinitely. A set check box indicates the Segment can repeat infinitely. A reset check box indicates the Segment cannot repeat infinitely. When the check box is set, the Maximum Repeat column is disabled. The first Segment within a Loop must not repeat infinitely.

EDI standard transaction

Table 10. Add EDI Standard Transaction Detail editor fields (continued)

Field	Description
Loop ID	The ID of the group of related EDI Standard Segments if the Segment is part of a Loop. By convention, a Loop is the same as a segment group in EDIFACT, ODETTE, or TRADACOMS EDI Standards. The Loop Level and Loop ID fields are used to determine the Segments that are part of a specific Loop.
Maximum Loop Repeat	This value indicates the maximum number of times the Loop can repeat. The value can be from 0 to 999,998. This field is disabled if the Unlimited Loop Repeat column is set. A value must be entered only for the first Segment within a Loop. The first Segment within a Loop must have the Maximum Repeat column set to 1. A value of zero in this column is treated as one.
Unlimited Loop Repeat	This check box is used to indicate that the Loop can repeat infinitely. A set check box indicates the Loop can repeat infinitely. A reset check box indicates the Loop cannot repeat infinitely. When the check box is set, the Maximum Loop Repeat field is disabled.
Loop Level	The Loop Level indicates the current nesting level of a Loop. Segments that are not within a Loop must have a value of zero in this column. When a Loop starts, the Loop Level for each Segment at that level of the Loop must have a value of 1. If a Loop is nested within that first Loop, then that nested Loop Level must be 2 to indicate the second level of looping. All Segments that are part of nested Loop must have the Loop Level of 2. Nested Loops can be up to 16 levels deep. The Loop Level and Loop ID columns are used together to determine the Segments that are part of a specific Loop.
Description	This column describes the Segment. This value in this column cannot be changed.

8. At this point you have two options:
 - To add the transaction and continue adding transactions, click Insert and return to step 7 on page 77.
 - To add the transaction and close the editor, click OK. The transaction is added to the list window and the editor closes.
9. Add any notes required. You must have a transaction defined before you can add a note. To add a note to an EDI standard transaction:
 - a. Select the **Notes** tab and click New. The Add EDI Standards Transaction Note editor opens.
 - b. Select the table and position that identifies the segment being referred to by this note.
 - c. Select from the Note Type list the code that specifies the type of this note.
 - d. Type the text of your note in the Note field.
 - e. At this point you have two options:
 - To add the note and continue adding notes, click Insert and return to step 9b.

- To add the note and close the editor, click OK. The note is added to the list window and the editor closes.
10. Select the Comments tab and enter any additional information about the transactions.
 11. When you have completed entering information, click Save on the tool bar to save the transaction.

Creating a segment

Use the EDI standard segment editor to enter new segments into an EDI standard or to edit existing segments. From the segments editor, you can add or edit the usage of data elements in segments.

The segments editor contains four tabs:

- The **General** tab contains the fields for you to name the segment and select its dictionary.
- The **Details** tab contains the fields for you to add or change data elements associated with the selected segment.
- The **Notes** tab displays a list window of the notes associated with specific segments. You can also add a new note from this list window.
- The **Comments** tab contains a field for you to type any comments.

The following instructions are for creating a new segment. For information about viewing, copying, editing, renaming, deleting, and printing segments, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new segment when business needs require one.

1. Select the Segments tab and click New on the tool bar.
The EDI Standard Segment editor opens with the **General** tab in front.
2. Type a name in the Segment field.

Notes:

 - a. The name is displayed in numbers and capital letters.
 - b. You cannot type spaces within the name.
 - c. The name can be up to 8 characters long and can contain alphanumeric characters and any of the special characters “!@#%&*()_+ =':;<,> .? ”.
3. Select the dictionary in which you want the segment to display through the Dictionary list.
4. Enter a description of the segment in the Description field, and a brief summary of the segment's purpose in the Purpose field. These fields are optional.
5. Select the **Details** tab and click New to enter information about the data elements contained in this segment.
6. Complete the Position and Element fields and any of optional fields you need. The following table describes the fields and their values.

EDI standard segment

Table 11. Add EDI Standard Transaction Segment editor fields

Field	Description
Position	The value in this field indicates the relative position of the Data Element within the Segment. Each value must be unique. The values do not need to be sequential. When the Segment is saved the list of Data Elements are sorted based on the position number. The valid values for this field are 1 to 32,000.
Element	This field is used to identify the name of the Data Element. Select a name from the list of available Data Elements.
Requirement Designator	Use this field to indicate whether the Data Element is optional, mandatory, or conditional. The conditional value indicates that the Data Element might be present depending on semantic conditions. Use the EDI Standard Notes action to see a list of Notes or to maintain Notes for the Segment.
Maximum Repeat	This value indicates the maximum number of times the Data Element can occur at this position. The value can be from 0 to 32,000.

7. At this point you have two options:
 - To add the segment and continue adding segments, click Insert and return to step 6 on page 79.
 - To add the segment and close the editor, click OK. The segment is added to the list window and the editor closes.
8. Add any notes required. You must have a segment defined before you can add a note. To add a note to a EDI standard segment:
 - a. Select the **Notes** tab and click New. The Add EDI Standards Segment Note editor opens.
 - b. Specify the position that identifies the Data Element that this note refers.
 - c. Select the Note Type from the list.
 - d. Select from the Relation list the code that specifies the relationship of the identified EDI standard data element to other EDI standard data elements listed in the position fields.
 - e. Select the position that identify up to ten related Data Elements to which the selected EDI Standard Data Element is related.
 - f. Type the text of your note in the Notes field.
 - g. At this point you have two options:
 - To add the note and continue adding notes, click Insert and return to step 8b.
 - To add the note and close the editor, click OK. The note is added to the list window and the editor closes.
9. Select the Comments tab and enter any additional information about the transactions.
10. When you have completed entering information, click Save on the tool bar to save the segment.

Creating a data element

Use the EDI standard data element editor to enter new data elements into a standard or to edit existing data elements.

The data elements editor contains four tabs:

- The **General** tab contains the fields for you to name the data element and select its dictionary.
- The **Details** tab contains the fields for you to add or change the component data elements associated with a composite data element.
- The **Notes** tab displays a list window of the notes associated with specific segments. You can also add a new note from this list window.
- The **Comments** tab contains a field for you to type any comments.

The following instructions are for creating a new data element. For information about viewing, copying, editing, renaming, deleting, and printing data elements, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Create a new data element when business needs require one.

1. Select the **Data Elements** tab and click New on the tool bar.
The EDI Standard Data Element editor opens with the **General** tab in front.
2. Type a name in the Data Element field.

Notes:

 - a. The name is displayed in numbers and capital letters.
 - b. You cannot type spaces within the name.
 - c. The name can be up to 8 characters long and can contain alphanumeric characters and any of the special characters “!@#%&*()_+ =';<, > . ?”.
3. Select the dictionary in which you want the data element to display through the Dictionary list.
4. Enter a description of the data element in the Description field, and a brief summary of the data element's purpose in the Purpose field. These fields are optional.
5. Select the data element's data type from the list in the data type field. The data types are explained in Table 13 on page 83.
6. Enter the minimum and maximum length of the data element in the Min Length and Max Length fields. These are required fields.
7. If the data element you are creating can only contain certain values, select the code list which identifies acceptable values for the data element you are creating from the Code List field.
If none of the existing code lists specify the acceptable values for your data element, you can create a new code list. See “Creating a code list” on page 84.
8. The **Details** and **Notes** tabs are only available when the data type is CD (composite data element).
 - If you are creating a composite data element and selected CD in the data type list, go to step 9 on page 82.

EDI standard data element

- If you are creating any other type of data element, go to step 13.
9. Select the **Details** tab to identify the component data elements that are associated with this composite data element and the properties of those associations.
 10. Complete the Position and Element fields and any of optional fields you need. The following table describes the fields and their values.

Table 12. Add EDI Standard Transaction Segment editor fields

Field	Description
Position	The value in this field indicates the relative position of the component Data Element within it's parent composite Data Element. Each value must be unique. The values do not need be sequential. When the composite Data Element is saved the list of component Data Elements is sorted based on the position number.
Element	This field is used to identify the name of the component Data Element. Select a name from the list of available Data Elements.
Requirement Designator	Use this column to indicate whether the component Data Element is optional, mandatory, or conditional. The conditional value indicates that the component Data Element might not be present depending on semantic conditions. Use the EDI Standard Notes action to see a list of Notes or to maintain Notes for the composite Data Element.

11. At this point you have two options:
 - To add the data element and continue adding data elements, click Insert and return to step 10.
 - To add the data element and close the editor, click OK. The data element is added to the list window and the editor closes.
12. Add any notes required. You must have a data element defined before you can add a note. To add a note to a EDI standard data element:
 - a. Select the **Notes** tab and click New. The Add EDI Standards Data Element Note editor opens.
 - b. Specify the position that identifies the Data Element that this note refers.
 - c. Select the Note Type from the list.
 - d. Select from the Relation list the code that specifies the relationship of the identified EDI standard data element to other EDI standard data elements listed in the position fields.
 - e. Select the position that identify up to five related Data Elements.
 - f. Type the text of your note in the Notes field.
 - g. At this point you have two options:
 - To add the note and continue adding notes, click Insert and return to step 12b.
 - To add the note and close the editor, click OK. The note is added to the list window and the editor closes.
13. Select the Comments tab and enter any additional information about the transactions.
14. When you have completed entering information, click Save on the tool bar to save the data element.

Table 13. Data types for EDI standard data elements

Data Type	Name	Description
A	Alphabetic	Alphabetic characters up to the length of the field.
AN	Alphanumeric	You can use any combination of characters in the ALPHANUM code list, up to the length of the field.
CD	Composite data element	A data element with data type CD and data element ID beginning with a C by convention for EDI standard composite data elements and S for envelope composite data elements. The component data elements are defined using the Details tab in the EDI Standard Data Element editor.
CH	Character	Any combination of characters up to the length of the field.
DT	Date	Date format yyymmdd, where yyyy is the year, mm is the month (01-12), and dd is the day.
ID	Identifier	A data element which usually has a code list for the valid values for the data element. For example, data element UM, unit of measure, has data type ID, and the valid values for this data element are listed in the UMCODES code list. The table name is the same as or starts with the data element ID.
IV	Incrementing value	A data element, such as a message reference number, that starts at 1 and increases by 1 for each usage. Note: IV is used for send and Receive maps only.
N	Numeric	Any combination of 0–9 and an optional sign (+ or -). The length includes the sign. When mapping data elements defined as data type N in UN/EDIFACT EDI standards, replace it with data type R (because data type N in the ASC X12 EDI standards is the same as data type R in UN/EDIFACT EDI standards).
Nn	Numeric	Numeric data with N places to the right of an implied decimal point. The only acceptable characters are the digits 0 through 9. N is the same as N0. For example, if the data type is N2, the value 123 means 1.23. A sign (+ or -) is optional. Positive is assumed if no sign is present. The length does not include the sign. This data type must be used when defining data elements defined as data type N in UN/EDIFACT EDI standards.
PW	Password	A password used in the interchange or functional group header.
R	Real	Numeric data that requires a decimal point for fractional values. The decimal point is optional for integers. A sign (+ or -) is optional. Positive is assumed if no sign is present. The length does not include the decimal point and sign. This data type must be used when defining data elements defined as data type N in UN/EDIFACT EDI standards.
Rn	Real	Signed or unsigned numeric data with a minimum of n significant decimal places. On sending, at least n decimal places are generated. On receiving, the decimal places are the same as data type R. A sign (+ or -) is optional. Positive is assumed if no sign is present. The length does not include the decimal point and sign.

EDI standard data element

Table 13. Data types for EDI standard data elements (continued)

Data Type	Name	Description
TM	Time	Time format is hhmm or hhmmss, depending on the length of the data element, where hh is the hour, mm is the minutes, and ss is the seconds. The time format uses a 24-hour clock, where the hour is specified as 01 to 24 for EDIFACT and 00 to 23 for X12.

Creating a code list

A code list is a list of acceptable values for data elements which can only contain certain values.

The following instructions are for creating a new code list. For information about viewing, copying, editing, renaming, deleting, and printing profiles, *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. For information about exporting profiles, *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Note: Code lists can also be specified in data format fields (when used with send and Receive maps) and can be used in all maps.

Create a new code list when you create a data element which can only contain certain values. A code list can also be created if you wish to restrict the values of any data item during translation. In this case, the code list is specified during mapping.

See “Creating code lists” on page 89 for the process to create a code list.

Editing an envelope standard

An envelope standard is a name under which components of an EDI envelope standard are grouped.

Note: Envelopes are used for send and Receive maps only.

The following procedure detail how to edit an envelope standard. For information about viewing, copying, editing, renaming, deleting, and printing profiles, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Envelope standards and control strings from the database are not used with Data Transformation maps. Instead the server uses plug-in enveloper and deenveloper modules for the supported standards.

Note: An envelope standard cannot be created - it must be imported. Obtain envelope standards from the WebSphere Data Interchange Web site at <http://www.ibm.com/websphere/datainterchange>.

Edit existing envelope standards when you want to change the default values of envelope fields for envelopes being created during translation.

1. Select the envelope standard you want to edit from the list displayed on the Envelope Standards list window.

The Envelope Standard editor opens with the **General** tab in front and the Dictionary field filled in with the name of the selected dictionary.

2. Edit the description of the envelope standard in the Description field.

Note: Click the Segments, Composite Elements and Simple Elements buttons to display list windows that contain the components associated with this envelope standard.

3. Enter the version and release of the envelope standard.
4. Select an Industry Code from the list provided; if a list is not available, you can type a name in the list. The available codes are:

RAIL Association of American Railroads

UCS Uniform Communication Standard

VICS Voluntary Inter-Industry Communications Standard

5. Select an Agency Code from the list provided; if a list is not available, you can type a name in the list. The available codes are:

T TDCC, ODETTE

UN UN/EDIFACT

X ASC X12, RAIL, UCS, VICS

6. Enter a purpose for the envelope.
7. Click the **Delimiters** tab to add the envelope specific information. Complete the fields on the **Delimiters** tab. The following table describes the fields and their values.

Table 14. Add EDI Standard Transaction Segment editor fields

Field	Description
Segment ID Separator	Segment ID separator is the character, or its hexadecimal value, that separates the segment ID and the first EDI standard data element in a EDI standard segment. To use an equal sign, for example, either select the character =, or select 7E (EBCDIC) or 3D (ASCII) from the list.
Segment Delimiter	Segment delimiter is the character, or its hexadecimal value, which marks the end of a EDI standard segment in an EDI standard transaction set. To use an exclamation point, for example, either select the character !, or select 5A (EBCDIC) or 21 (ASCII) from the list.
Data Element Delimiter	Data Element delimiter is the character, or its hexadecimal value, that separates EDI standard data elements in an EDI standard transaction set. To use the asterisk, for example, either select the character *, or the value 5C (EBCDIC) or 2A (ASCII) from the list.

EDI envelope standard

Table 14. Add EDI Standard Transaction Segment editor fields (continued)

Field	Description
Subelement Delimiter	Subelement delimiter is the character, or its hexadecimal value, that separates subelements (EDI standard composite data elements) in a EDI standard transaction set. To use a colon, for example, either select the character :, or select 7A (EBCDIC) or 3A (ASCII) from the list.
Release Character	Release character is the character, or its hexadecimal value, which indicates where a delimiter character is part of the data. To use quotation marks, for example, either select the character quotation marks ("), or select 7F (EBCDIC) or 22 (ASCII) from the list. Assuming that the asterisk ordinarily separates EDI standard data elements, an asterisk that is part of the data must, for this example, be preceded by quotation marks ("*") to be recognized as data.
Decimal Notation	Decimal Notation is the character that represents the decimal point in numeric values of an EDI standard transaction set. To use a dot, for example, select the character dot (.) from the list. The EDIFACT standard uses comma as the primary decimal mark.
Optional Functional Groups	Optional Functional Groups is a check box which indicates whether or not the functional group envelope is optional. A selected check box indicates that functional groups are optional. An cleared check box, indicates functional groups are required.
Delimiter Segment ID	Delimiter Segment ID is the ID of the EDI standard segment used to send your delimiter values to a trading partner. An example is the UNA segment for EDIFACT.
Interchange Header	Interchange Header is the segment ID of the interchange header for this envelope standard. The value cannot be changed for this standard.
Interchange Trailer	Interchange Trailer is the segment ID of the interchange trailer for this envelope standard. The value cannot be changed for this standard.
Group Header	Group Header is the segment ID of the functional group header for this envelope standard.
Group Trailer	Group Trailer is the segment ID of the functional group trailer for this envelope standard.
Transaction Header	Transaction Header is Enter the segment ID of the EDI standard transaction set header for this envelope standard.
Transaction Trailer	Transaction Trailer is the segment ID of the EDI standard transaction set trailer for this envelope standard.

8. Select the **Comments** tab and add any additional information about the envelope.
9. Click Save on the tool bar to save the changes to the envelope standard.

To view lists of EDI standard components in the current envelope standard:

1. Click the button in the List Of group box corresponding to the component you wish to view.

A list window displaying EDI standard components associated with this envelope standard displays:

- The Segments button opens the EDI Standard Segments list window.

- The Composite Elements button opens the EDI Standard Data Elements list window showing all composite data elements in this envelope standard.
 - The Simple Elements button opens the EDI Standard Data Elements list window showing all simple data elements (those that are not composite data elements) in this envelope standard.
2. You can open any of the components by double-clicking them.

Envelope control strings

An envelope control string helps improve performance of the translator when envelopes are prepared. Envelope control strings must be compiled by WebSphere Data Interchange Client after they are installed and after any change is made. If standalone mode is being used on the WebSphere Data Interchange Client, the envelope control string must be exported from the WebSphere Data Interchange Client and imported into the server. In client/server mode, the envelope control string is automatically stored on the server database during the compile.

The Envelope Control String list window is opens a list of compiled envelope standards and shows the compilation information about each.

To compile an envelope control string:

1. Select the envelope control string to be compiled from the list displayed on the Envelope Control Strings list window. Alternately, select the envelope standard on the Envelope Standards list window.
2. Click Compile Control String(s) on the tool bar of the list window.
While compiling, WebSphere Data Interchange Client checks for errors in the changes you made. Error messages are written to the event log.

EDI envelope control string

Chapter 7. Creating map objects

Use map objects to customize your maps. The following steps describe how to access the map objects.

1. Click Mapping to open the Mapping functional area.
2. Select one of the following tabs:
 - Code Lists
 - Translation Tables
 - Global Variables

Creating code lists

You create a new code list when you need to ensure a value is valid during translation. For instance, you might want to ensure that a simple element contains a valid part number. The following steps describe how to create a new code list.

1. Open the Code Lists editor. The following steps describe how to do this.
 - a. Click either Mapping or EDI Standards on the tool bar of the main application window.
 - b. Select the Code Lists tab, click New on the list window tool bar. The Code List editor opens.
2. On the **General** tab of the Code List editor, type in the name of the code list into the Name field. This value must be unique among translation tables and code lists.
3. In the Description field, type a description of the code list. This field is optional.
4. In the Data Type field, indicate whether the values contained in the code list are character or numeric. Character is the default.
5. In the Maximum Length field, identify the maximum length of the values. The length can be up to 35 characters long. 5 is the default.
6. Click New to open the Add New Code List Entry dialog to create the code list entries. Each entry identifies a valid value within the code list.
7. In the dialog, enter a value that is to be considered valid and its corresponding description.
8. Click Insert to insert the entry. The dialog remains open so another entry can be made. Click OK to add the last entry to the code list, or click Cancel if the last entry has already been made.
9. Click the Comments tab, and type any comments you have about the code list.
10. Click Save on the toolbar. WebSphere Data Interchange saves the new code list to the database.
11. Close the editor when you are done.

Creating translation tables

You create a new translation table when you need to translate a value in a document to a corresponding value. For instance, a simple element that contains an internal part number that must be converted to a trading partner's corresponding part number.

The following steps describe how to create a new translation table.

1. Open the Translation Table editor. The following steps describe how to do this.
 - a. Click Mapping on the main application window.
 - b. In the Translation Tables tab, click New. The Translation Table editor opens.
2. On the **General** tab of the Translation Table editor, type in the name of the translation table in the Name field.
3. You can optionally type a description of the translation table in the Description field.
4. In the Source Value group box, indicate whether the source values are character or numeric in the Data Type field.
5. In the Source Value group box, identify the maximum length of the source values in the Maximum Length field. The length can be up to 35 characters long.
6. In the Target Values group box, indicate whether the target values are character or numeric in the Data Type field.
7. In the Target Value group box, identify the maximum length of the target values in the Maximum Length field. The length can be up to 63 characters long. The combined length of the source and target values cannot exceed 68 characters.
8. Click New to open the Add New Translation Table Entry dialog to create the translation table entries. Each entry associates a source value with a target value. In the dialog, enter a source value and its corresponding target value. Click Insert to insert the entry. The dialog remains open so another association between a source value and a target value can be made. Click OK to add the last entry to the translation table, or click Cancel if the last association has already been made.
9. Click the Comments tab, and add any comments you have about the translation table.
10. Click Save on the toolbar. WebSphere Data Interchange Client saves the new translation table to the database.
11. Close the editor when you are done.

Creating global variables

Create a new global variable when you need a variable for a Data Transformation map, validation map, or Functional Acknowledgement map that needs to retain its information beyond the current document being translated.

The following steps describe how to create a new global variable.

1. Open the Global Variables editor. The following steps describe how to do this.
 - a. Click Mapping on the main application window.
 - b. Select the Global Variables tab, click New. The Global Variable editor opens.

2. On the **General** tab, type in the name of the global variable into the Name field.
3. You can type a description of the variable in the Description field. This field is optional.
4. Indicate the scope of the global variable as either Session, Interchange, or Group.
5. From the Data Type list, select the data type of the global variable. The choices are: Binary, Boolean, Character, Integer, and Real. The default is character.
6. In the Maximum Length field, you can set the maximum length of the global variable. The default is 32.
7. In the Initial Value field, type an initial value for the global variable.
8. Click Save on the toolbar. WebSphere Data Interchange saves the new global variable to the database.
9. Close the editor when you are done.

Global variables

Chapter 8. Creating a map

Mapping is the process by which you relate input and output documents. Through mapping, you specify the relationships between the internal documents used by your applications and the external documents that you exchange with trading partners. Maps are then compiled using WebSphere Data Interchange Client into control strings so that translation servers can transform documents between the internal and external formats.

WebSphere Data Interchange Client is designed to make mapping easy. You can choose to select elements in your source document, drag them onto elements in the target document, and drop them. Or you can choose to select elements in your target document, drag them onto elements in the source document, and drop them. You can then apply WebSphere Data Interchange's specialized mapping functions as required.

Using the map editor

WebSphere Data Interchange Client's map editor features a visual means for associating elements from a source document definition with elements in a target document definition, or from a target document definition with elements in a source document definition.

The editor contains three tabs:

- The **General** tab contains the fields for you to enter and change map properties.
- The **Details** tab contains the fields for you to create and maintain the mapping commands
- The **Comments** tab contains a field for you to type any comments you wish about the selected map.

The **Details** tab uses a split page that permits you to create map associations. The divisions on this page varies based on the type of map.

- For Data Transformation maps, the **Details** tab has four window panes as follows:
 - Upper left corner
This is the Source Document Definition pane of the **Details** tab. This pane identifies the source document definition and displays the layout of the document that is used as the source of the translation.
 - Upper right corner
This is the Target Document Definition pane of the **Details** tab. This pane identifies the target document definition and displays the layout of the document that is created by the translation.
 - Lower left corner
This is the Mapping Command pane of the **Details** tab. This pane displays a representation of either the source document definition or the target document definition, depending on whether the map is a source-based or target-based, and the mapping commands and comments associated with the map.
 - Lower right corner

This is the Variables pane of the **Details** tab. This pane displays the variables associated with the map. This pane is divided into three sub-panes that list all special variables, global variables, and local variables.

- For Validation maps, the **Details** tab has two window panes as follows:
 - Left side

This is the Mapping Command pane of the **Details** tab. This pane displays a representation of the source document definition. The representation is similar to the source document definition, but it might not be exactly the same. Use this pane to create and maintain mapping commands.
 - Right side

This is the Variables pane of the **Details** tab. This pane is divided into three sections, one for each type of variable. Use the Variables window to manage Local Variables and to map with Global Variables, Local Variables, and Special Variables.
- Functional Acknowledgment maps have four window panes as follows:
 - Upper left corner

This is the Source Document Definition pane of the **Details** tab page. This pane identifies the source document definition and displays the layout of the document that is used as the source.
 - Upper right corner

This is the Target Document Definition pane of the **Details** tab page. This pane identifies the target document definition and displays the layout of the document that is created.
 - Lower left corner

This is the Mapping Command pane of the **Details** tab page. This pane displays a representation of the source document definition. The representation is similar to the source document definition, but it might not be exactly the same. Use this pane to create and maintain mapping commands.
 - Lower right corner

This is the Variables pane of the **Details** tab. This pane displays the variables associated with the map. This pane is divided into three sub-panes that list all special variables, global variables, and local variables.
- Send maps have two window panes as follows:
 - Left side

This is the Source Document Definition pane of the **Details** tab. This pane displays a representation of the source document definition. The representation is similar to the source document definition, but might not be exactly the same.
 - Right side

This is the Mapping pane of the **Details** tab. This pane displays a representation of the document definition that describes the layout of the target document.
- Receive maps have two window panes as follows:
 - Left side

This is the Target Document Definition pane of the **Details** tab. This pane displays a representation of the target document definition. The representation is similar to the target document definition, but might not be exactly the same.

- Right side

This is the Mapping pane of the **Details** tab. This pane displays a representation of the document definition that describes the layout of the source document.

Starting the map editor

The Map editor opens when you select a map from the Mapping list window, as follows.

1. Click Mapping on the WebSphere Data Interchange Client Navigator bar.

The Mapping Functional Area opens.

This window is used to access the list windows for each component type that occurs in the Mapping Functional Area.

To display additional columns, click the scroll bar on the bottom of the page to scroll to the right or left. To alter the columns that display on the page, or to change which query is executed to produce the list, click Modify Window Properties. To create new queries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

2. To view a map or to add or change its information, double-click the row of the map you want to work with.

The map editor opens with the **Details** tab in front. You add information or make changes to maps through its tabs and related windows, as described in the following sections.

General editing procedures

Using the **Details** tab in the Map editor you can perform drag-and-drop mapping on your documents.

General editing procedures

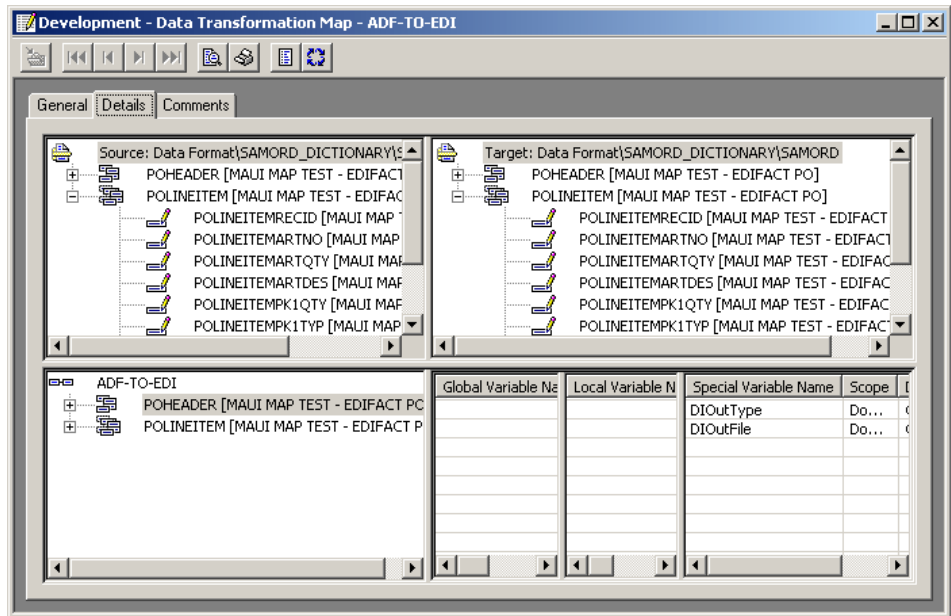


Figure 8. Data Transformation Map editor, Details tab

The top left pane in the window opens the source document definition, and the top right pane displays the target document definition. The lower left pane is the Mapping Command window pane, and the lower right pane is the variables window pane, which includes the lists for Global, Local, and Special Variables.

For more information about the Map Command window pane, see “Using the Map Command window pane” on page 99.

For a list of mapping components and their associated graphics, see Table 15 on page 98.

1. Click the plus (+) sign next to a compound element, such as a loop, segment, or record in the target document definition. Simple elements, such as fields and data elements, do not have a plus (+) sign next to them.
The compound element expands to show the compound and simple elements that comprise the compound element.
If you wish to close any expanded compound element, click the minus (-) sign.
2. Click the element you want to map on the top left side of the page. While holding down the mouse button, drag it to the corresponding element in the target document definition on the top right pane.

As you drag the element to the right side of the page over the element with which you want to associate it, that component becomes highlighted. Release the mouse button.

You can drag all simple elements and repeating compound elements. When you are dragging an element, holding it over a compound element causes the compound

element to expand after a few moments, if it is not already expanded. Dragging an element to any edge of the window pane causes the window pane to automatically scroll up, down, left or right, depending on which edge you are near and whether the window pane can be scrolled in that direction.

Note: If the element does not become highlighted when you move the cursor on top of it, that means it is not a valid place to perform a drop. For example, you cannot drop a data format record onto a data element in an EDI transaction.

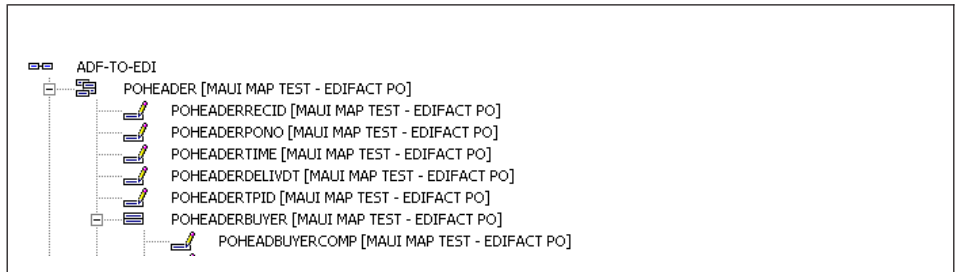


Figure 9. Example of a Data Transformation Map

After you have mapped an element, a command is created in the mapping commands window pane and inserted into an appropriate position. A check mark displays next to the mapped element in the mapping command window. A check mark also displays next to its parent elements so that you know the parent element contains mapped components.

Note: This procedure presents a simple mapping situation in which you associate elements in the source document definition with elements in the target document definition. For anything but the most basic mapping associations, you must use WebSphere Data Interchange's advanced mapping capabilities. For more information, "Using the Map Command window pane" on page 99

3. Continue dragging and dropping elements in the source document definition onto elements in the target document definition until you have mapped all of the information required for this map.
4. Enter any general comments on the map in the Comments tab.
5. When you have completed mapping, click Save on the tool bar to save the map.

Editing a map

Edit a map when you have modified the associated target or source document definition, or when you need to add, delete, or change mapping commands within the map. You might also need to edit a map to meet specific requirements of a new trading partner.

1. In the Mapping list window, double-click the map you wish to edit.
The map displays in the map editor with the **Details** tab in front.

General editing procedures

Changes you made earlier to the map or its associated items display on the page, as well as changes you made to the source or target document definition.

2. Create new associations between the source and target elements, or change existing ones.
 - To make an association between a new element in the source document definition and an element in the target document definition, drag the source element and drop it on the proper target element.
 - To edit an existing association, double-click the mapping command in the Mapping Command window pane. The Mapping Command editor opens.
 - To delete an association, select the mapping command you want to delete and press the Delete key.
3. Change information as required in the General tab.
4. Click Save on the tool bar to save the map.

Table 15. Symbols used in maps























Symbol	Mapping	Data Format	EDI Standard	XML DTD
				
			EDI Standard Transaction	
			Table	
		Record	Segment	
		Repeating Record	Repeating Segment	
		Loop	Loop	
		Repeating Loop	Repeating Loop	
		Structure	Composite Data Element	Compound Element
		Repeating Structure	Repeating Composite Data Element	Repeating Compound Element
		Field**	Simple Data Element**	Simple Element**
			Repeating Simple Data Element**	
	Mapping Indicator			

Table 15. Symbols used in maps (continued)

Symbol	Mapping	Data Format	EDI Standard	XML DTD
	Mapping Command			
	Command Group			
	Comment			
	Comment Group			
	If, Elself, Else, EndIf			
	Qualify			
	Qualify (multi-occurrence)			

** Simple Elements - all others are Compound Elements

Using the Map Command window pane

The Map Command window pane is where you can apply WebSphere Data Interchange Client's advanced mapping capabilities.

1. Right-click an element in the Mapping Command window pane and follow the cascading menus. You can choose to insert a command, a command group, a comment, or a comment group. You can choose to insert your selection before, after, or within the element you have selected.

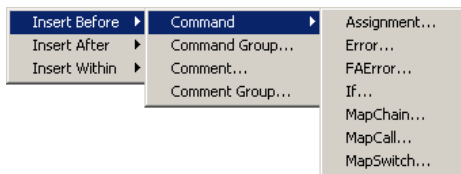


Figure 10. Cascading menus in the Map Command window pane

2. If you want to insert a command, select where the command is inserted, select Command, and then select the command to be inserted.
The Mapping Command editor opens with a prototype of the command.
3. Edit the prototype to create the command you need.
4. Click Repeat if you want to add the command, and create another similar to it.
5. Click OK when you have your last command ready.

Advanced mapping techniques

WebSphere Data Interchange techniques for using literal keywords and other advanced mapping techniques are documented in Chapter 14, “Advanced send and Receive mapping,” on page 241.

To make mapping easier, WebSphere Data Interchange Client help contains syntax for literals, mapping commands, and operators.

Mapping task list

Follow this process to create a map. The same basic steps work for maps that you create to translate data that are sent to trading partners and for maps that you create to translate data that are received from trading partners. In both cases, you are mapping data from one document format to another.

1. Obtain the layout of the source document to be used in the translation.

This might be a data format, an EDI standard transaction, or an XML DTD. You might need to obtain the layout from your trading partner. When you have the source document layout (called the source document definition), put that layout into WebSphere Data Interchange.

If the source document is data in a proprietary format, create a data format in WebSphere Data Interchange. For information about creating a data format, see “Creating a data format” on page 26.

If the source document is an EDI standard transaction, study the layout of the transaction. EDI standards are broad. You can use a number of methods to map onto an EDI standard transaction. Any number of organizations are likely to use different methods to map the same data to the same transaction. When you are comfortable with the transaction, make sure the transaction is defined in WebSphere Data Interchange. For instructions on how to install EDI standards, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

If the source Document is in XML format, obtain the corresponding DTD and import it into WebSphere Data Interchange. This is done using import. For information about importing a DTD file, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Attention: If you change the metadata for a document, it affects each map that uses that document as either a source or target document.

2. Obtain the layout of the target document to be used in the translation.

This might be a data format, EDI standard transaction, or an XML DTD. You might need to obtain the layout from your trading partner. When you have the target document layout (called the target document definition), put that layout into WebSphere Data Interchange. See step 1 for additional information.

3. Decide how source data is handled.

When you are creating a map to send a document to a trading partner, study the target document definition and decide how you want to use it to pass your source data.

When you are creating a map that receives a document from a trading partner, study your trading partner's document definition to see what data your partner is sending and how you handle it.

4. Compare your source document definition to the target document definition.
When you map, you are associating elements in one document definition with elements in another document definition.
Study the source document definition for which you are creating a map. Make note of which elements in the source document definition correspond to which elements in the target document definition. Then decide how you are going to map the two document definitions. For example, you might use a loop on an EDI standard transaction to handle repeating records in a data format.
5. Map the data elements in each segment.
Use the Map editor to drag elements between the source and target document definitions. You might notice that some elements in target document definition do not occur in the source document definition. If they are required, you need to fill them with data on the outbound side. WebSphere Data Interchange can fill many such elements using special handling options, literals, mapping commands, and accumulators.
Special handling options and mapping commands perform such functions as translating date formats, converting values supplied by a trading partner to values you require, and validating the contents of data elements.
Accumulators and variables perform such actions as counting segments in a transaction for later placement in a transaction's trailer record.
Literals and mapping commands are a means of supplying data to data elements or fields, such as dates and times.
6. Specify how repeating simple and compound elements are handled.
Document definitions can include repeating elements. Examples of repeating elements are loops in a data format, records, and structures. You must specify how WebSphere Data Interchange handles the repeating elements.
7. Associate the map with a trading partner using a usage or map rule.
After you have created a map, you must associate it with trading partners. Because you can use the same map for multiple trading partners, each usage or rule identifies one or more trading partners that use the map. The usage or rule can supply values specific to that trading partner or group of trading partners, such as:
 - Validation and error levels
 - Unique values for enveloping that override the default values
 - The logical name of the file to which output data is writtenFor more information, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.
8. Compile a map.
The WebSphere Data Interchange uses a control string in its processing, not the map itself. When you complete your mapping work, you must compile the map using WebSphere Data Interchange Client to create a control string. The compiler identifies any errors when it creates a control string.

Choosing the right map

WebSphere Data Interchange supports several types of maps that you can use to transform data that you exchange with other business and trading partners:

- Data Transformation maps

Using Data Transformation maps, you can define any-to-any transformations for creating complex mapping operations. These support the wide range of mapping that you might need to handle data formats, EDI, and XML messages.

For information about creating a Data Transformation map, see Chapter 9, “Data Transformation mapping,” on page 121.

- Functional Acknowledgement maps

A set of standard Functional Acknowledgement maps is sent with WebSphere Data Interchange. You can customize these maps, or create your own if you have special requirements for your functional acknowledgements.

For information about creating a Functional Acknowledgement map, see “Creating a Functional Acknowledgement map” on page 137.

- Validation maps

Validation maps specify extended checking of the EDI input data, beyond the normal standard validation performed by WebSphere Data Interchange. Using a validation map, you can check whether the data conforms to an implementation guide or other user-defined requirements.

For information about creating a validation map, see “Creating a validation map” on page 132.

- Send and Receive maps

Send and Receive maps are supported in Version 3.3 of WebSphere Data Interchange for migration and compatibility. If you have created send and Receive maps in previous releases, you can continue to use these.

For information about creating send and Receive maps, see Chapter 13, “Send and Receive mapping,” on page 201.

Specifying qualification

Terminology

You must understand the following terminology when you use qualification.

Simple element

A single value

Compound element

A grouping of elements

Element

Refers to either a simple or compound element

In data formats, simple elements are fields within the data format. Compound elements include structures, records, and loops.

In EDI standards, simple elements are data elements or subelements. Compound elements include loops, segments, and composite data elements.

An XML element is a compound element. An XML attribute and an XML value are considered simple elements.

Within your document, certain elements and groups of elements can repeat. When you map an element that repeats within the source document (whether compound or simple element), you must tell WebSphere Data Interchange which occurrence of the segment or loop you are using. This is called *qualifying* the segment or loop.

WebSphere Data Interchange supports five types of qualification for maps. You can qualify by:

- Occurrence
- Multi-occurrence (Data Transformation maps only) or path (send and Receive maps only)
- Value
- Expression

Mapping tip

When creating new maps with complex looping, start mapping a few fields in the loop and testing the qualification before mapping the entire document. It is easier to test and correct the looping with a subset of fields before completing the entire Map.

When you qualify by occurrence, your mapping is related to the position of data in a repeating sequence. When you qualify by value, your mapping is related to the value that you receive in a simple element or in a variable. When you qualify by expression, your mapping is based upon a condition. For multi-occurrence qualification, you are

Qualification

indicating what mapping instructions are executed for iterations of the repeating element that are not handled in a previous qualification

Note: You can mix types of qualification for the same loop, segment, or data element. Path qualification works with Value qualification on receive. Occurrence works with Path on both send and receive. You cannot mix Occurrence and Value qualification.

Qualifying a Data Transformation map

WebSphere Data Interchange supports four types of qualification for Data Transformation maps. You can qualify by:

- Occurrence
- Multi-occurrence
- Value
- Expression

When you qualify by occurrence, your mapping is related to the position of data in a repeating sequence. When you qualify by value, your mapping is related to the value that you receive in a simple element or in a variable. When you qualify by expression, your mapping is based upon a condition. For multi-occurrence qualification, you are indicating what mapping instructions are executed for iterations of the repeating element that are not handled in a previous qualification. If there are no qualifications in place for a repeating element, then it is handled as a multi-occurrence qualification by default.

If there are no qualifications in place for a repeating element, then it is handled as a multi-occurrence qualification by default. However, when Overlays occur in the Target data, a **CloseOccurrence** mapping command might be necessary and placement is extremely important. There is a WebSphere Data Interchange feature with Target based maps that executes Source looping structures multiple times. For example, you can execute mapping commands for all occurrences of the loop and execute a different set of mapping commands for all occurrences of the same Loop. This can be beneficial if you want to save some values from the source looping structure and then execute the source looping structure again and use the saved values.

Qualifying repeating simple and compound elements

Compound elements can be qualified by single occurrence, multi-occurrence, value, and expression.

Qualify by occurrence when the order in which repeating data occurs in either the source or target document is important. For more information, “Qualifying an element by occurrence” on page 105

Qualify by multi-occurrence when you want to create multiple elements in the target to correspond to repeating elements in the source using the same mapping instructions. All iterations of the repeating element that are not otherwise qualified are handled under the multi-occurrence qualification. For more information, see “Qualifying an element by multi-occurrence” on page 106

Qualify by value when you want a set of mapping instructions to be executed for a repeating element when a specific value occurs in a source element or in a variable. For more information, see “Qualifying an element by value” on page 107

Qualify by expression when you need a more complex qualification. For more information, see “Qualifying an element by expression” on page 108

Qualifying an element by occurrence

Qualify an element by occurrence when a specific instance of a repeating element requires mapping instructions specific to that occurrence of the repeating element. For example, say that you are working with a source document that has an address compound element. The first occurrence of the address compound element has the send-to address, the second occurrence has the ship-to address. Map each occurrence to the appropriate target element using occurrence qualification.

Note: Whether you are doing qualification in a source or target-based map, you are always qualifying on something in the source. `Qualify(Occurrence() = 1)` means the first Occurrence of something in the source, not an Occurrence of something in the target.

1. For a source based map, right-click the repeating source element in the Mapping Command window pane. For a target based map right-click the **ForEach** command underneath the repeating target element. If one does not exist, then first right-click the repeating target element and select **ForEach** to create one. For more information about the ForEach command see “Qualifying an element by multi-occurrence” on page 106.
2. Select By Occurrence.
The Qualify by Occurrence Number window opens.
3. Type in a number in the Enter the Occurrence Number field that corresponds to the occurrence number of the element you are mapping.
If you are mapping the second occurrence of an element, type 2 in the Enter the Occurrence Number field.
4. If you need to map additional occurrences, click Repeat.
A mapping command is created that qualifies the element by occurrence and then redisplay the Qualify by Occurrence Number window.
5. When you have specified the required number of occurrences, click OK.
A mapping command is created that qualifies the element by occurrence, and the Qualify by Occurrence Number window is closed.

You can qualify an element by more than one occurrence in a single expression. For example the expression:

```
Qualify ((Occurrence() EQ 1) OR (Occurrence() EQ 2))
```

performs the same set of mapping commands for occurrence one and occurrence two of the repeating element. This command can be created by using the **Qualify by expression** command or by creating a qualify by occurrence and then editing the qualify by occurrence command after it has been created.

Qualifying an element by multi-occurrence

Qualify an element by multi-occurrence when you need WebSphere Data Interchange to handle all occurrences of a repeating element, that are not otherwise qualified, in the same way. For example, say you are working on an EDI standard transaction to data format map and you find that the PO1 segment in a purchase order repeats to handle multiple purchase-order line items. You need WebSphere Data Interchange to create a separate record for each instance of the PO1 segment and when each occurrence of the PO1 segment is to execute the same mapping instructions.

Consequently, you would qualify the PO1 segment by multi-occurrence. That way, WebSphere Data Interchange creates as many line-item records in your application data as there are occurrences of PO1 in your trading partner's transaction.

The WebSphere Data Interchange mapping function automatically qualifies repeating elements. The mapping commands under the repeating element is executed for each occurrence of the repeating element.

The multi-occurrence qualification commands for source-based maps and target-based maps are different. Source-based maps use **Mapto()** and the Default qualification commands, but target-based maps use the **ForEach** and **Default** commands. In addition, you can only have one multi-occurrence qualification (Default) on a source repeating element, but you can have multiple multi-occurrence qualifications (**ForEach**) on a target repeating element. Although they have different names and usage rules, they accomplish the same thing.

For example, if you want to map two repeating source elements (S1 and S2) to a single repeating target element (T1), then you either create two Default qualifications in a source-based map (one on S1 and one on S2), or two **ForEach** qualifications in a target based map (both on T1).

The main difference between multi-occurrence qualification in source-based and target-based maps is that because you can map more than one repeating source element to a repeating target element, the tree display has an extra level of hierarchy. In a source based map the qualifications appear directly under the repeating source element. In a target based map, each repeating source element associated with the repeating target element results in the creation of a **ForEach** command under the repeating target element. The qualifications related to that source element appear under the **ForEach** command.

If there are no qualifications in place for a repeating element, then it is handled as a multi-occurrence qualification by default. However, overlays can occur in the Target data and a **CloseOccurrence** mapping command might be necessary and placement is extremely important. There is a WebSphere Data Interchange feature for target based maps that executes source looping structures multiple times. For example: You can execute mapping commands for all occurrences of the loop and execute a different set of mapping commands for all occurrences of the same Loop. This can be beneficial if you want to save some values from the source looping structure and then execute the source looping structure again and use the saved values.

In source-based maps, the multi-occurrence **Default** command only displays if there are other existing qualifications for the same repeating element. When you are using multiple qualifications, multi-occurrence qualification always displays last when it is present. This indicates that mappings under the multi-occurrence qualification are performed only when the specific instance of the repeating source element is not resolved to one of the other qualifications. This is why it is known as the default qualification.

In target-based maps, the multi-occurrence qualification **ForEach** command displays whenever there is any type of qualification on the element. Default only displays when there are other existing qualifications for the same repeating source element. When multiple qualifications exist and the multi-occurrence qualification is present, it always displays last. This indicates that mappings under the multi-occurrence qualification are performed only when the specific instance of the repeating source element is not resolved to one of the other existing qualifications. This is why it is known as the default qualification.

Drag a compound element onto a repeating element in the source document definition:

- If there are no existing qualifications for the element, a **Mapto()** command is inserted under the corresponding repeating element in the Mapping Command window pane. For source based maps, if there are no existing qualifications for the element, a **Mapto()** command is inserted under the corresponding repeating source element in the Mapping Command window pane. For target based maps, if there are no existing qualifications for the element, a **ForEach** command is inserted under the corresponding repeating target element in the Mapping Command window pane.
- If there is an existing qualification for the repeating element, the **Default** command is inserted as the last qualification for the repeating element in the Mapping Command window pane.

The multi-occurrence command **Default** is automatically created when you first qualify a repeating element that already has mapping commands under it. The existing mapping commands are moved to under the **Default** command.

Qualifying an element by value

Qualify an element by value when you want the value of data in a simple element or variable to drive WebSphere Data Interchange's translation of a repeating element.

For example, say you want to qualify the N1 loop with the value of BY in Element 98, which is the "Entity Identifier Code," received in a purchase order to create a buyer record. Further, say that you want the buyer's name to be mapped into the buyer record depending on the value in Element 98 of the N1 loop.

To handle that case, you would qualify the element by value. That way, WebSphere Data Interchange puts specific information into the buyer records it creates from the purchase order depending on the name in each order's Entity Identifier Code.

1. For a source based map, right-click the repeating source element in the Mapping Command window pane. For a target based map, right-click the **ForEach** command underneath the repeating target element. If one does not exist, right-click the repeating target element and select **ForEach** to create one.

Qualification

2. Right-click to select Qualify.
3. Click By Value.
The Mapping Command editor opens.
4. Drag the element you wish to qualify from the source document window and drop it onto the word **path** on the Mapping Command editor.
5. Highlight the word **value** in the Qualify field in the Mapping Command editor.
6. Type the value.
7. If you are creating multiple qualifications by value, click Repeat. Then repeat the previous steps.
Clicking Repeat creates the qualification in the Mapping Command window pane and redisplay the Mapping Command editor.
8. When you have created the required number of qualifications of this element, click OK.
Clicking OK creates the qualification in the Mapping Command window pane and closes the Mapping Command editor.

You can qualify an element by several different values in a single expression. For example the expression:

```
Qualify ((StrComp(\Table 1\310 0 N1 Loop\310 0 N1\1 M 98\, "BY") EQ 0) OR  
(StrComp(\Table 1\310 0 N1 Loop\310 0 N1\1 M 98\, "BG") EQ 0))
```

performs the same set of mapping commands if the value in element 98 is "BY" or "BG"

Qualifying an element by expression

1. For a source based map, right-click the repeating source element in the Mapping Command window pane. For a target based map, right-click the **ForEach** command underneath the repeating target element. If one does not exist, right-click the repeating target element and select **ForEach** to create one.
2. Right-click to select Qualify.
3. Click By Expression.
The Mapping Command editor opens.
4. Enter a valid expression into the Mapping Command editor.
5. If you are creating multiple qualifications by expression, click Repeat. Then repeat the previous steps.
Clicking Repeat creates the qualification in the Mapping Command window pane and redisplay the Mapping Command editor.
6. When you have mapped the required number of qualifications of this element, click OK.
Clicking OK creates the qualification in the Mapping Command window pane and closes the Mapping Command editor.

Editing an element qualification

Edit a qualified element when you want to change the qualification.

If the element is qualified by occurrence, value, or expression, and you wish to change occurrence, value, or expression qualifications:

1. Double-click the qualify command in the Mapping Command window pane that is to be changed.
The Mapping Command editor opens.
2. Edit the qualification.
3. Click OK.

Changing multi-occurrence qualification on a qualified element

1. Create a multi-occurrence qualification if one does not already exist.
2. Check that the multi-occurrence qualification command (**Default**) is selected.
3. Drag repeating element from the source document definition onto a repeating element in the target document definition.
A **MapTo** command is created in the Mapping Command window pane under the multi-occurrence qualification command (Default).
4. If the new **MapTo** command is to replace an existing **MapTo** command, delete the existing **MapTo** command by right-clicking on it and selecting Delete.

Qualifying a repeating data element or composite data element by occurrence

Qualify a repeating data element or composite data element by occurrence when a specific instance of the data element must be mapped to a specific component area of the data format.

1. Double-click the repeating data element or composite data element (maximum use value greater than one).
The Qualify an Element by Occurrence window opens.
2. Type in a number in the Enter the Occurrence Number field that corresponds to the occurrence number of the data element you are mapping.
If you are mapping the second occurrence of the data element, type 2 in the Enter the Occurrence Number field.
3. If you need to map additional occurrences, click Repeat.
The qualification is added to the map and displays on the transaction side of the Map editor. The Qualify an Element by Occurrence Number window redisplay with the next available number in the Enter the Occurrence Number field increased by one.
4. When you have specified the required number of occurrences, click OK.
The qualification is added to the map and displays on the transaction side of the Map editor.

Qualifying a repeating data element or composite data element by path

1. Drag a structure onto a repeating data element (a data element that has a maximum use value greater than one).
The title of the data element or composite data element changes to include the words Qualified by Path of (Path Name).

Changing the path qualification on a qualified repeating data element

1. Drag a structure onto the loop or repeating segment.
For Send maps, the Qualify an Element window opens. Move to step2.
For Receive maps, you can only have one path qualified mapping for a repeating data element. Dragging a structure onto an existing path qualified repeating data element results in the qualification being replaced by the new path qualification.
2. When the Qualify an Element window opens, click New or Replace, depending on whether you want to add another qualification or replace the existing qualification.

Qualifying a send or Receive map

Within the EDI standards, certain segments and groups of segments, called loops, can repeat. Some EDI standards also support data elements or composite data elements that can repeat.

When you map a segment, loop, or data element that repeats within an EDI standard, you must tell WebSphere Data Interchange which occurrence of the segment, loop, or data element you are using. This is called qualifying the segment, loop, or data element.

WebSphere Data Interchange supports three types of qualification for send and Receive map types. You can qualify by:

- Occurrence
- Path
- Value (loops and repeating segments on Receive maps only)

Non-repeating data elements can also be qualified by value on Receive maps.

When you qualify by occurrence, your mapping is related to the position of data in a repeating sequence. When you qualify by path, your mapping is related to a specific structure or record in the data format and how it handles repeating segments or data elements in the EDI standard. When you qualify by value, your mapping is related to the value that you receive in a data element.

Qualifying loops and segments

You can qualify loops and segments by:

- Occurrence when the order in which repeating data occurs in either the data format or EDI standard transaction is important. For more information, see “Qualifying a loop or segment by occurrence” on page 111.
- Path when you want to create multiple segments in an EDI standard transaction to correspond to multiple occurrences of a record or structure in a data format and vice versa. For more information, see “Qualifying a loop or segment by path” on page 111.
- Value when you want to specify data received in a standard transaction to trigger the creation or population of fields in your data format. For more information, see “Qualifying a loop or segment by value” on page 112.

Qualifying a loop or segment by occurrence

Qualify a loop or segment by occurrence when a specific instance of the segment or loop must be mapped to a specific component of the data format. For example, say that you are working on a Send map and need to send two addresses, the send-to address and the bill-to address. Map the ship-to address to the first occurrence of the N1 segment and the bill-to address to the second occurrence of N1.

Attention: Qualification by occurrence creates records or segments first and then looks for the data to fill them. If you use occurrence qualification on a Receive map, then you need to move data to that record because WebSphere Data Interchange creates the record, and it might not contain any data.

WebSphere Data Interchange Client's mapping function automatically qualifies loops and segments that repeat. When you drop a field onto a data element that occurs in a loop or a repeating segment, the loop or segment is automatically qualified by occurrence; the title of the loop or segment changes after the drop to include the words, "Qualified by Occurrence #1". Use the Qualify a Loop or Segment window, as follows.

To qualify a loop or segment by occurrence:

1. Double-click a loop or repeating segment (a segment that has a maximum use value greater than one).

For Receive maps, either the Qualify a Loop or Qualify a Segment window opens. For Send maps, proceed to step 3.

Note: You can also drop a field onto a data element in a loop or repeating segment. The loop or segment is automatically qualified by occurrence. See "Editing a loop or segment qualification" on page 113 to change the occurrence number.

2. Click Occurrence.

Either the Qualify a Segment by Occurrence window or the similar window for loops displays. The name of the loop or segment displays in the title bar.

3. Type in a number in the Enter the Occurrence Number field that corresponds to the occurrence number of the component you are mapping.

If you are mapping the second occurrence of a field in your data format to an EDI standard transaction, type 2 in the Enter the Occurrence Number field.

4. If you need to map additional occurrences, click Repeat.

The qualification is added to the map and displays on the transaction side of the Map editor. The Qualify a Segment by Occurrence Number window or its corresponding loop window redisplay with the next available number in the Enter the Occurrence Number field.

5. When you have specified the required number of occurrences, click OK.

The qualification is added to the map and displays on the transaction side of the Map editor.

Qualifying a loop or segment by path

Qualify a loop or segment by path when you need WebSphere Data Interchange to create multiple instances of either:

Qualifying

- a record or structure for Receive maps
- a segment or loop for Send maps

For example, say you are working on a Receive map and find that the PO1 segment in a purchase order repeats to handle multiple purchase-order line items. You need WebSphere Data Interchange to create a separate record for each instance of the PO1 segment.

Consequently, you would qualify the PO1 segment by path. That way, WebSphere Data Interchange creates as many line-item records in your application data as there are occurrences of PO1 in your trading partner's transaction.

Note: Qualification by path finds loops or repeating segments and creates records for them on receive and vice versa on send. Use path qualification when the number of records WebSphere Data Interchange might need to create is unknown.

WebSphere Data Interchange Client's mapping function automatically qualifies loops and segments that repeat. When you drop a record or structure onto a loop or repeating segment, the loop or segment is automatically qualified by path; the title of the loop or segment changes after the drop to include the words, "Qualified by Path of (Path Name)," as follows.

To qualify a loop or segment by path:

1. Drag a record or structure onto a loop or repeating segment (a segment that has a maximum use value greater than one).
The title of the loop or segment changes to include the words, "Qualified by Path of (Path Name)."

Qualifying a loop or segment by value

Qualify a loop or segment by value when you want the value of data received in a data element to drive WebSphere Data Interchange's translation of a whole loop or segment. Qualification by value is not supported on Send maps.

For example, say you want to qualify the N1 loop with the value of BY in Element 98 (Entity Identifier Code) received in a purchase order to create a buyer record. Further, say that you want the buyer's name to be mapped into the buyer record depending on the value in Element 98 of the N1 loop.

To handle that case, you would map the segment by value. That way, WebSphere Data Interchange puts specific information into the buyer records it creates from the purchase order depending on the name in each order's Entity Identifier Code.

To qualify a loop or segment by value in Receive maps:

1. Double-click a loop or a repeating segment (a segment that has a maximum use value greater than one).
The Qualify a Loop or Qualify a Segment window opens.
2. Click Value.

Either the Qualify a Segment by Element window or a similar window for loops opens. The name of the segment or loop displays in the title bar.

3. Select from the Select an Element list the name of the data element by which you want to qualify the loop or segment. The list includes all data elements in the segment or in the first segment within a loop.
4. In the Enter a Qualifying Value list, select a value or type in the value you want to qualify the loop or segment on. The list contains values when a code list is associated with the selected data element.
5. If you are mapping multiple occurrences, click Repeat.
The qualification is added to the map and displays on the transaction side of the Map editor. Then either the Qualify a Segment by Element window or the similar window for loops redisplay. Repeat Step 4 and Step 5.
6. When you have mapped the required number of occurrences of this segment, click OK.
The qualification is added to the map and displays on the transaction side of the Map editor.

Editing a loop or segment qualification

Edit a qualified loop or segment when you want to change the default qualification that WebSphere Data Interchange Client places on a segment when you use drag-and-drop mapping of fields, structures, and records.

To edit a qualified loop or segment:

If the Loop or Repeating Segment is Path Qualified and you wish to change to or add occurrence or value qualifications:

1. Double-click the qualified loop or repeating segment.
The Qualify a Segment window opens.
2. Click New or Replace, depending on whether you want to add another qualification or replace the existing qualification.
If this is a Receive map and you have no other qualifications for this loop or repeating segment, the Qualify a Loop or Qualify a Segment window opens with the choices Value, Occurrence, and Cancel. Continue with step 3.
If other qualifications exist for the loop or segment, then additional qualifications must be of the same type. The Qualify a Loop by an Element, Qualify a Segment by Element, Qualify a Loop by Occurrence Number, or Qualify a Segment by Occurrence Number window opens directly in this case.
For Send maps, the Qualify a Loop by Occurrence Number or Qualify a Segment by Occurrence number displays. Skip to step 4.
3. Click Occurrence or Value (if given the choice) depending on how you want to qualify the loop or segment.
4. When you have qualified the required number of occurrences of the loop or segment, click OK.

Changing path on qualified repeating element

1. Drag a record or structure onto the loop or repeating segment.
The Qualify a Segment or Qualify a Loop window opens.
2. Click New or Replace, depending on whether you want to add another qualification or replace the existing qualification.

Note: For Receive maps, you can only have one path qualified mapping for a loop or segment. You are issued an error message if you attempt to create a second path qualified mapping.

Qualifying data elements

Nonrepeating data elements can be used to qualify by value other nonrepeating data elements. Repeating data elements can be qualified by occurrence number or path. A repeating data element can be qualified by both an occurrence number and a path.

Qualify data elements by value when you receive data elements that have qualifiers in a segment. In such cases, you might need to qualify how WebSphere Data Interchange handles each occurrence of the data element. Data element qualification by value is not supported on Send maps.

If data you receive from a trading partner contains many units of measure, (each, case, for example), you might need to qualify each data element. By doing so, WebSphere Data Interchange can translate data in the data elements into a single unit of measure for your application.

When you qualify a repeating data element by occurrence, your mapping is related to the position of data in a repeating sequence. When you qualify a repeating data element by path, your mapping is related to a specific structure in the data format and how it handles repeating data elements in the EDI standard.

Qualifying a data element by value in Receive maps

Use the Add an Element Qualification window to qualify a data element by the value of another data element, as follows.

1. Double-click the data element in the segment that will be used to qualify one or more data elements in the segment. Make sure you qualify data elements before you map them.
The Qualified Element Support window opens.
2. Click Qualified.
The Add an Element Qualification wizard opens.
If you click Normal, the Mapping Data Element editor opens. For more information, see "Using the mapping data element editor" on page 202
3. Select the data element or data elements you want to qualify.
 - a. Select the data element or data elements in the Select Elements group box.
 - b. Click > to move the selected data element or data elements to the Qualified Elements group box.
Clicking >> moves all data elements in the list.

Clicking < or << removes the selected element or all data element(s) from the Qualified Elements group box.

4. Click Next.

The second page of the Add an Element Qualification wizard opens.

5. Select or enter a qualifying value. Values appear in the list if a code list is associated with the qualifying element.

a. Click the value or values from the Enter a Value list. You can also type a value in the list.

b. Click > to move the value to the Qualifying Values group box.

Clicking >> moves all values in the list.

Clicking < or << removes the selected value or all values from the Qualifying Values group box.

6. Click Finish.

An element mapping icon (or icons, if you selected several values) displays below the data element or data elements you qualified and below the qualifying data element. The mapping element is designated Not Mapped - Qualified by Element in Position x with a Value of y where x is the position of the qualifying data element in the segment and y is the value you selected.

Editing qualification of a data element qualified by value

1. Double-click the qualifying data element if you want to add additional values.

The Update an Element Qualification window opens.

2. Select or enter a new qualifying value. Values appear in the list if a code list is associated with the qualifying element.

a. Click the value or values from the Enter a Value list. You can also type a value in the list.

b. Click > to move the value to the Qualifying Values group box.

Clicking >> moves all values in the list.

Clicking < or << removes the selected value or all values from the Qualifying Values group box.

3. If you want to see all information about the qualified element, click Previous Selected Info.

The Previous Element Qualification Data window opens, displaying all data elements qualified by this element and the values used to qualify those elements.

4. Click OK.

An element mapping icon (or icons, if you specified several values) displays below the data element or data elements you previously qualified. The mapping element is designated Not Mapped - Qualified by element in Position x with a Value of y where x is the position of the qualifying element in the segment and y is the value you specified.

Compiling control strings

After you complete a map and associate it with trading partners, you must compile a control string before WebSphere Data Interchange can use the map. WebSphere Data Interchange Client uses the data you created during the mapping process as input to a program that compiles the map to create a control string. The WebSphere Data Interchange Server uses the control string in its translation processing.

While compiling, WebSphere Data Interchange Client checks for errors in the map you created. Error messages are displayed in the Execution Status window. Serious errors are also logged to the Message log.

Compiling a control string is the last thing you do after adding or updating a map. Any time you change a map, you must compile a new control string.

To compile a control string:

1. Select the tab for the map type.
2. In the map list window, select the maps you want to compile.
3. Click Compile on the list window tool bar.

An Execution Status window opens. Any errors are noted in the window.

Note: If you are not using WebSphere Data Interchange Client in stand alone mode, export the control string into the WebSphere Data Interchange Server. For more information about exporting and importing, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Compiled control strings display in the Control Strings list window. Click Mapping on the navigator bar to view the control strings list window. Then click the **Control Strings** tab to view the Control Strings list window. Table 16 describes the fields that display in the control strings list window.

Table 16. Control string list window field descriptions

Field	Description
Map Name	The name of the map that this control string compiled.
Generation Date	The date the control string was compiled.
Generation Time	The time the control string was compiled.

Recompiling control strings

Map control strings are recompiled using WebSphere Data Interchange Client any time the corresponding map is changed or any time the related source or target document definition is changed. Changes are not available to WebSphere Data Interchange until a recompile has been successfully completed.

The following steps describe how to recompile map control string.

1. Open the Control Strings list window by clicking Mapping on the WebSphere Data Interchange navigator bar and selecting the Control Strings tab.
2. Select one or more map control strings in the Control Strings list window.
3. Click Compile on the toolbar, or select Compile from the Actions menu.

WebSphere Data Interchange Client recompiles the selected control strings and opens an Execution Status window. When WebSphere Data Interchange is finished recompiling, a message appears on the Execution Status window indicating that the compile has been completed. You can click Cancel on the Execution Status window to terminate the compile function.

Migrating a map

The source or target document definition in a map can be changed at any time. This is most commonly done to change from one version of a document definition to another. This occurs commonly with EDI standards. To change the source or target document definition:

1. On the **General** tab of the map editor, select a different source or target document from the menus. When changing the version or release of an EDI standard transaction, usually only the dictionary name must be changed. For more information about using the map editor, see “Using the Data Transformation Map editor” on page 121.
2. Click Save.

For Data Transformation maps, the map editor tries to validate existing mapping commands. If the base document definition was changed, the map editor attempts to locate the appropriate place in the Mapping Commands window pane to migrate the existing mapping commands. Any mapping commands that cannot be migrated display with a red exclamation mark. You can edit those mappings or drag them to the correct place in the map. Any elements in the source document definition that the map cannot identify appear with a red exclamation mark. Any commands under these can be moved or deleted as needed. When you are sure you no longer need the unknown elements that have a red exclamation mark by them, you can delete them by right-clicking and selecting Delete.

Migrating a map

Part 2. Data Transformation Maps

Chapter 9. Data Transformation mapping

A Data Transformation map is a set of mapping instructions that describes how to translate data from a source document into a target document. The order in which the mapping instructions occur can be based on the source document (source-based mapping) or the target document (target-based mapping). Both the source and target documents can be one of several supported document types.

This chapter contains the following:

- “XML mapping considerations” on page 123
- “Applying map rules” on page 125
- “Mapping MQMD and MQRFH2 values” on page 126

Using the Data Transformation Map editor

You can use the Data Transformation Map editor to show how data is to be translated from the source document to the target document. The editor works by displaying the source document definition on one side of the page and target document definition on the other. Using the Data Transformation Map editor you can associate components of your source document with components of the target document by dragging source components and dropping them into the correct locations in the target document definition. Components include simple elements, such as fields in a data format or data elements in an EDI standard transaction, and compound elements, such as loops and record in a data format and segments in an EDI standard transaction.

You can map the simple elements in the source document definition to the simple elements in the target document definition in any of these patterns:

- One simple element to one simple element
- Several simple elements to one simple element
- One simple element to several simple elements

The Data Transformation Map editor contains three tabs:

- The **General** tab contains the fields for you to enter and change map properties.
- The **Details** tab contains the fields for you to create and maintain the mapping commands.
- The **Comments** tab contains a field for you to type any comments you wish about the selected map.

The following section is the procedure for creating a Data Transformation map. For information about viewing, copying, editing, renaming, deleting, and printing maps, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. For information about exporting maps, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Creating a new Data Transformation map

Create a new map when you want to translate a document from one format to another.

In the most basic sense, creating a new map consists of associating simple elements in a source document definition with simple elements in a target document definition. The following procedure shows how WebSphere Data Interchange makes basic associations between simple elements.

1. In the Mapping list window, click the Data Transformation maps tab.
2. Click New on the tool bar.
The Create a Data Transformation Map—Map Name wizard page opens.
3. Type a map name in the Map Name field.
You can use both letters and numbers to identify your map. Letters display in capitals. You cannot type spaces within the name.
You can enter a more complete description of the map in the Description field.
4. Click Next to continue. The Source or Target wizard opens.
Select the appropriate radio button according to whether you want to create a source-based Data Transformation map, or a target-based Data Transformation map.
5. Click Next to continue.
The Source Syntax Type wizard opens.
6. Select the appropriate radio button that indicates the syntax type of your source document definition, and click Next to continue.
The Source Dictionary wizard opens with a list of the dictionaries for your specified syntax type.
7. Select your source dictionary and click Next.
A wizard opens with a list of the document definitions in your source dictionary. The wizard page is titled Source Data Format, Source EDI Standard Transaction, or Source XML DTD, depending on the Source Syntax Type previously selected.
8. Select the data format, the EDI Standard transaction, or the XML DTD to be used as the source document definition in your map, and click Next.
The Target Syntax Type window opens. In this window you can indicate the syntax type of your target document definition.
9. Select the syntax type of your target document, and click Next.
The Target Dictionary wizard opens with a list of the dictionaries for your specified syntax type.
10. Select your target dictionary, and click Next.
A wizard opens with a list of the document definitions in your target dictionary. The wizard page is titled Target Data Format, Target EDI Standard Transaction, or Target XML DTD, depending on the Target Syntax Type previously selected.
11. Select the data format, the EDI standard transaction, or the XML DTD to be used as the target document definition in your map, and click Next.
The Confirmation window opens.
12. Confirm the selections. If they are correct, click Finish to save the information.

After finishing the map information, the Map editor is opened with the **Details** tab in front.

- a. Click the **General** tab.

The **General** tab contains general information about the map. It includes the map name, a brief description of the map and it identifies the source and target document definitions.

The name of the map is set when the map is created. It displays on the **General** tab for informational purposes only. If you want to change it, you must use the rename action.

Use the Description field to provide a brief description of the map. The description can be up to 50 characters long.

Mapping hierarchical loops

A hierarchical loop (HL) is similar to an organization chart. Just as an organization chart shows you the various groups of people and their relationships to the whole, a hierarchical loop shows you each group of data and its relationship to the whole.

Hierarchical loops define different levels of data, which can be used in any sequence and skipped when appropriate. By defining hierarchical loops you can place the loop anywhere in your data.

Note: WebSphere Data Interchange includes support for Hierarchical Loops. For detailed explanations of HL loops and how WebSphere Data Interchange handles them, see Appendix B, "Hierarchical loops," on page 297.

XML mapping considerations

With Namespaces you can use multiple XML schema definitions within an XML document or building a grammar from several different schemas, by providing a way to resolve name conflicts between the schemas. For example, one schema might define an address element to be a person name, that includes street, city, state, and zipcode elements. Another might define address to be an e-mail address that contains a simple string. By using different namespaces, these can be uniquely identified.

For example, to show that element address belongs to a particular namespace, a schema or instance document defines a prefix for the namespace. For example, it might define `xmlns:po="http://example.com/ns/P0Example"`. Then when the qualified element appears in the data, the element would appear as `<po:address>`, to show that it is part of that namespace. The `po` prefix in this example is just a shorthand way to represent the namespace `"http://example.com/ns/P0Example"`. Even though the schema might use prefix `po`, instance documents might use different prefixes, for example `mypo`, `purch`, or even define it as the default namespace, so no prefix is used for this element. As long as these were defined to the `"http://example.com/ns/P0Example"` namespace in the instance documents they must all be considered equivalent.

The mapping and translation for XML schemas is *namespace aware*. This means that it recognizes that the prefix represents a namespace, and gives it special treatment

Advanced mapping techniques

instead of just treating it as part of the element name. Since the schema translation and mapping is namespace-aware, elements with the same name and namespace would be considered to be equivalent, even if they used different prefixes.

Namespaces

The **Namespaces** tab provides information that is associated with each namespace URI. Each namespace table entry is associated with an XML dictionary. Besides the namespace URI and dictionary name, each entry includes the following information for the URI when it is used in that dictionary.

- Description
This is a text description for the namespace. It is only used to help the user identify the namespace, and is not used for mapping or translation.
- Prefix
This is the prefix that will be used for the namespace when qualified elements and attributes are displayed in the map, and when the elements and attributes are written in the XML output. If no prefix is specified, or if no namespace entry exists for a namespace URI, then the element or attribute is written without a prefix.
- Schema location - This specifies the schema location that is used if the namespace URI is used in a **SetSchemaLocation** command.

When a schema is imported, the schema is scanned for any `xmlns` attributes. If an `xmlns` attribute is found namespace for a namespace URI that is not already defined for the dictionary, a new namespace table entry is created using the prefix defined for the attribute. The description and schema location can be added later if needed.

The prefix and schema location are not included in the internal map representation, so the map does not need to be recompiled if these values are changed.

Target Namespace

The target namespace tells which namespace a schema describes. It is identified in the schema by the `targetNamespace` attribute.

When a schema is imported, the schema is scanned for a `targetNamespace` attribute. If this attribute is found, the target namespace for the schema is set from the attribute value. The target namespace associated with the schema can be changed, but typically must not be unless the import was not able to find it correctly. Setting the target namespace to a value that does not match the `targetNamespace` attribute can prevent WebSphere Data Interchange from parsing the schema correctly, which results in errors when trying to map the document.

Namespace Processing for Input XML documents

You must specify `XMLNS(Y)` on your **PERFORM** command if you are doing schema validation, or your XML schema uses namespace qualified elements/attributes. Namespace processing is required for schema validation, because the attributes that help locate the schema definitions (`xsi:schemaLocation` and `xsi:noNamespaceSchemaLocation`) use namespace qualification. If your source or

target document is a schema that uses qualified elements or attributes, namespace processing is required so the internal names used by the translation process matches the internal names used in the map.

Namespace Processing for Output XML Documents

When generating output XML documents, qualified elements and attributes use the prefix specified by the namespace table entry for the URI. If no prefix is specified, or if no namespace entry exists for a namespace URI, then the element or attribute is written without a prefix.

Some special attributes are sometimes needed in the XML output to identify the namespaces used and the schema location(s). Mapping commands that specify attributes are used for XML output only and are ignored for EDI and data format output.

XML schema restrictions

Some special content types that are defined by schemas have limitations on their mapping capabilities. These include:

- `xsd:anyType`
This means that this element can contain any type of content, including child elements, simple elements, or mixed content. In WebSphere Data Interchange can only be mapped as if it were a simple string element.
- `xsd:any`
This means that any child element can appear in this position, sometimes subject to namespace restrictions. In WebSphere Data Interchange you cannot map to or from this element.
- `xsd:anyAttribute`
This means that any attribute can appear on this element. In WebSphere Data Interchange you cannot map to or from this attribute.
- Substitution groups
This means that one element can be substituted for another. In WebSphere Data Interchange you cannot map to or from the substitution group elements.

Applying map rules

When you have completed a map, you must associate it with a trading partner or trading partners. WebSphere Data Interchange calls those associations rules, map rules, or Data Transformation map rules.

Applying the minimal trading partners concept

The concept of minimal trading partners attempts to reduce the amount of time spent on administrative functions. The traditional WebSphere Data Interchange was based on the idea that each trading partner would be identified to the product through a trading partner profile and a map rule or usage. Thus, a WebSphere Data Interchange installation with tens of thousands of trading partners would require an equal number of profiles and map rules or usages, even though the options might be identical. The WebSphere Data Interchange concept of generic rules and usages reduces the administrative impact of this model, but does not completely meet all its needs. Some

Minimal trading partners

installations do not need a setup for a trading partner because they keep that information in their application and pass it to WebSphere Data Interchange at transformation time. WebSphere Data Interchange uses a combination of techniques and terminology to accommodate this minimal administrative model.

The following procedures can be done from the trading partner list windows as well as from the Mapping list window. For more information about trading partners see the *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

Viewing map rules

To view a map rule:

1. In the Data Transformation Maps list window, select a map.
2. Click Usages and Rules on the tool bar.

WebSphere Data Interchange Client runs a query and opens a window containing the Data Transformation Map Rule list window. A list of map rules associated with the map displays.

Mapping MQMD and MQRFH2 values

This enhancement maps commands to get and set the values of the MQMD and MQRFH2 headers used by WebSphere MQ. By accessing the values in these MQ headers from your maps, WebSphere Data Interchange can be more easily integrated with other WebSphere MQ and JMS applications.

The header values from the input message can now be read using the **GetProperty** command. The header values on the output message can now be set using the **SetProperty** command.

Getting and setting properties in the MQMD and MQRFH2 headers

To get the values in the MQMD and MQRFH2 headers you use the **GetProperty** mapping function in your map. To set new values in these headers, you use the **SetProperty** command.

You must specify the fully qualified path, starting with ROOT. Each component of the path is specified by a dot (.). Unlike the other properties, this path name is case sensitive.

The MQMD properties are specified by path "ROOT.MQMD". For example, "ROOT.MQMD.ReplyToQ" is the ReplyToQ field in the MQMD header.

The MQRFH2 properties are specified by path "ROOT.MQRFH2". For example "ROOT.MQRFH2.Encoding" is the Encoding field in the MQRFH2 header. Folders within the MQRFH2 folder, such as the mcd and usr folders can also be specified as part of the path. For example, "ROOT.MQRFH2.mcd.Set" is the Set value in the mcd folder of the MQRFH2 header.

Sample mapping commands are:

- Get the value of the MQMD MsgId and save it in variable MyMsgId:

- ```
MyMsgid = GetProperty("ROOT.MQRFH2.MsgId")
```
- Get the value of the MQRFH2 Format field and save it in variable Rfh2Fmt:  
Rfh2Fmt = GetProperty("ROOT.MQRFH2.Format")
  - Get the value of the domain (msd) from the MQRFH2 mcd folder:  
MsgDomain = GetProperty("ROOT.MQRFH2.mcd.Msd")
  - Set the value of field "MyField" in the usr folder of the MQRFH2 header:  
SetProperty("ROOT.MQRFH2.usr.MyField", "My user data")

Some of the fields in the MQMD and MQRFH2 use integer or binary values, instead of the character values used by the GetProperty and SetProperty functions. To access these values:

- The GetProperty function converts the from character when you get MQMD and MQRFH2 fields.
- The SetProperty functions converts the to character when you set MQMD and MQRFH2 fields.

When you get these properties from the source message:

- Integer values are converted to the character representation.
- Binary values are encoded, similar to the HexEncode function. For example, an 8-byte binary value of x0123456789ABCDEF would be returned as a 16-character string 0123456789ABCDEF.
- Character values include the blank padding when they are read from fixed-length header fields.

When you set these properties in the target message:

- For integer values, the character string is converted to an integer.
- For binary values, the encoded character string must be passed, similar to the value passed to HexDecode function. For example, to set an 8-byte binary value of x0123456789ABCDEF you must pass a 16-character string 0123456789ABCDEF. If the string is too short, it is padded with null characters. If the string is too long, it is truncated. If unable to decode the string, a warning message is issued.
- For character values, it truncates the string or pad with blanks if needed for fixed-length fields.

The supported properties and associated types are listed in Table 17.

*Table 17. MQMD properties (ROOT.MQMD.xxx)*

| Name     | Type | Length | Description                 |
|----------|------|--------|-----------------------------|
| StruclD  | Char | 4      | Structure identifier        |
| Version  | Int  |        | Structure version number    |
| Report   | Int  |        | Options for report messages |
| MsgType  | Int  |        | Message type                |
| Expiry   | Int  |        | Message lifetime            |
| Feedback | Int  |        | Feedback or reason code     |

Table 17. MQMD properties (ROOT.MQMD.xxx) (continued)

| Name            | Type       | Length | Description                              |
|-----------------|------------|--------|------------------------------------------|
| Encoding        | Int        |        | Numeric encoding of message data         |
| CodedCharSetId  | Int        |        | Character set identifier of message data |
| Format          | Char       | 8      | Format name of message data              |
| Priority        | Int        |        | Message priority                         |
| Persistence     | Int        |        | Message persistence                      |
| MsgId           | Binary(24) | 24     | Message identifier                       |
| CorrelId        | Binary(24) | 24     | Correlation identifier                   |
| BackoutCount    | Int        |        | Backout counter                          |
| ReplyToQ        | Char(48)   | 48     | Name of reply queue                      |
| ReplyToQMgr     | Char(48)   | 48     | Name of reply queue manager              |
| UserIdentifier  | Char(12)   | 12     | User identifier                          |
| AccountingToken | Binary(32) | 32     | Accounting token                         |
| AppIdentityData | Char(32)   | 32     | Application data relating to identity    |
| PutAppType      | Int        |        | Type of application that put the message |
| PutAppName      | Char(28)   | 28     | Name of application that put the message |
| PutDate         | Char(8)    | 8      | Date when message was put                |
| PutTime         | Char(8)    | 8      | Time when message was put                |
| AppOriginData   | Char(4)    | 4      | Application data relating to origin      |

Table 18 shows properties for Windows and AIX only (not z/OS and CICS).

Table 18. MQMD properties for Windows and AIX

| Name           | Type   | Length | Description                                                      |
|----------------|--------|--------|------------------------------------------------------------------|
| GroupId        | Binary | 24     | Group identifier                                                 |
| MsgSeqNumber   | Int    |        | Sequence number of logical message in group                      |
| Offset         | Int    |        | Offset of data in physical message from start of logical message |
| MsgFlags       | Int    |        | Message flags                                                    |
| OriginalLength | Int    |        | Length of original message                                       |

Table 19. MQRFH2 properties (ROOT.MQRFH2.xxx)

| Name    | Type | Length | Description              |
|---------|------|--------|--------------------------|
| StrucId | Char | 4      | Structure identifier     |
| Version | Int  |        | Structure version number |

Table 19. MQRFH2 properties (ROOT.MQRFH2.xxx) (continued)

| Name           | Type | Length | Description                                                 |
|----------------|------|--------|-------------------------------------------------------------|
| StrucLength    | Int  |        | Total length of MQRFH2 including NameValueData              |
| Encoding       | Int  |        | Numeric encoding of data that follows NameValueData         |
| CodedCharSetId | Int  |        | Character set identifier of data that follows NameValueData |
| Format         | Char | 8      | Format name of data that follows NameValueData              |
| Flags          | Int  |        | Flags                                                       |
| NameValueCCSID | Int  |        | Character set identifier of NameValueData                   |

Values in MQRFH2 folders such as the mcd (ROOT.MQRFH2.mcd.xxx) and usr (ROOT.MQRFH2.usr.xxx) are treated as character. No padding or truncation is done.

## Other notes

- The ability to get/set the MQRFH2 values is supported on Windows, AIX, and z/OS. It is not supported on CICS. The ability to get/set the MQMD values is supported on all platforms.
- WebSphere Data Interchange still sets the following values in the MQRFH2 header as before: Encoding, CodedCharSetId, and the mcd folder values. So if the user sets any of these values, they get overwritten by WebSphere Data Interchange specified values.
- The updated MQMD/MQRFH2 is only used if EDIRFH2 is used as the network program. Network program EDIMQSR continues to use the original MQMD header as before (not the values set in the map), and does not use an MQRFH2 header.
- The MQMD and MQRFH2 headers are not saved in the Document Store. The header values cannot be retrieved or set in the map when doing deferred translation, deferred enveloping, or re-enveloping.
- Default values are used for any MQMD/MQRFH2 values that are not set by the user.
- The MQMD/MQRFH2 values set by the user are not validated by WebSphere Data Interchange. If the user sets these to invalid values, they can cause errors when the message is written to the queue or when it is received by another application.

## Map rules

---

## Chapter 10. Validation mapping

Validation maps provide the instructions needed to perform additional validation beyond what is specified in the EDI Standard. The name of the validation map is specified in the map rule when it is to be used to perform additional validation on the source or target document in a translation. Validation maps contain mapping commands that are instructions used to provide additional validation of an EDI Standard Transaction.

A validation map is used to handle extended validation requirements, and to call an extended error function to report the information needed to create a functional acknowledgement. A validation map can also copy to local variables, and use most other mapping functions. A in cannot produce an output.

This chapter contains the basic information for creating validation maps. For detailed information about mapping techniques see Chapter 9, “Data Transformation mapping,” on page 121, and Chapter 12, “Data Transformation mapping commands and functions,” on page 141.

---

### Starting the validation map editor

WebSphere Data Interchange Client's editor features a visual means for defining your extended validation criteria.

The editor displays a mapping commands window pane on its left side. The mapping commands window pane displays the layout of the document to be validated with mapping commands and comments inserted into it. These commands are used to check the source document for various user-defined conditions, and set appropriate errors or functional acknowledgement information if the conditions are not met. There is also a variables window pane in the map editor. It is divided into three sub-panes that list all special variables, global variables, and local variables.

The editor opens when you select a map from the Mapping list window, as follows.

1. Click Mapping on the WebSphere Data Interchange Client tool bar.

The Mapping Functional Area opens.

This window is used to access the list windows for each component type that occurs in the Mapping Functional Area. Select the **Validation Maps** tab to view the Validation Map list window. The list window shows all of the Validation Maps selected by the current query. Each row in a list window is opens information about a different Validation Map.

To display additional columns, click the scroll bar on the bottom of the page to scroll to the right or left. To alter the columns that display on the page, or to change which query is executed to produce the list, click the Modify Window Properties icon on the WebSphere Data Interchange Client Navigator bar. To create new queries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

2. To view a map or to add or change its information, double-click the row of the map you want to work with.

## The editor

The editor opens, with the **Details** tab in front. You add information or make changes to maps through its tabs and related windows, as described in the following sections.

---

### Creating a validation map

Create a new map when you want to handle extended validation requirements, or you want to call an extended error function to report the information needed to create a functional acknowledgement.

A validation map usually includes a number of **If** commands to check whether the document meets various user-defined conditions. If the conditions are not met, then the **Error** or **FAError** command is used to write a message to the log and set an error code for the transaction. The **FAError** command sets functional acknowledgement information, in addition to writing the log message and setting the error code. The following procedure shows how to create a validation map.

1. In the Mapping list window, click the Validation Maps tab.
2. Click New on the list window tool bar.

The Create a Validation Map window opens.

3. Type a map name in the Map Name field.

You can use both letters and numbers to identify your map. Letters display in capitals. You cannot type spaces within the name. A must have a unique map name. Click Show Existing Map Names to view a list of map names.

4. Enter a detailed description of the map in the Description field. This field is optional.

You can enter the later using the **General** tab of the Validation map editor.

5. Click Next to continue.
6. The Source dictionary opens. Select the EDI standard dictionary to use as the source document definition in the validation map you are creating.
7. Click Next to continue.
8. The Source EDI Standard Transaction opens. Select the EDI standard transaction to use as the source document definition for your map.
9. Click Next to continue.
10. The Confirmation window opens. Confirm the selections. If they are correct, click Finish to save the information.

After finishing the map information, the Map editor opens with the **Details** tab in front.

---

### Using validation maps

You can use Validation maps in the same way as you can Data Transformation Maps. Many of the concepts explained for Data Transformation maps in Chapter 9, "Data Transformation mapping," on page 121 also apply to Validation Maps. However, note the following:

- Validation maps are source-based maps, in which the commands are based on the order of the source document. Commands that are only valid for target-based maps are not permitted in a validation map.
- You can use the **FAError** command in Validation Maps. For information about **FAError**, see Chapter 12, “Data Transformation mapping commands and functions,” on page 141.
- Validation maps do not have a target document. Commands and functions that specify a target element, such as MapTo, are not permitted in a validation map.

For more information about Validation Mapping techniques and concepts, see the following sections in Chapter 9, “Data Transformation mapping,” on page 121:

- “Specifying qualification” on page 103
- “Specifying HL levels” on page 297
- “Applying map rules” on page 125
- “Compiling control strings” on page 116

WebSphere Data Interchange techniques for using literal keywords and other advanced mapping techniques are documented in Chapter 12, “Data Transformation mapping commands and functions,” on page 141.

## Creating a validation map



---

## Chapter 11. Functional Acknowledgement mapping

A Functional Acknowledgement map is a Data Transformation map that provides the instructions on how to produce a functional acknowledgement to be returned to your trading partner. After creating the Functional Acknowledgement map, you can select it on the trading partner rules setup as the map to be used during functional acknowledgement generation. Both source and target based mapping are available for Functional Acknowledgement maps.

For detailed information about mapping techniques see Chapter 9, “Data Transformation mapping,” on page 121, and Chapter 12, “Data Transformation mapping commands and functions,” on page 141

---

### Functional Acknowledgement maps provided with WebSphere Data Interchange

Several Functional Acknowledgment maps are provided with WebSphere Data Interchange:

*Table 20. Standard Functional Acknowledgement maps*

| Map name        | Description                                                  |
|-----------------|--------------------------------------------------------------|
| &DT_FA997V2R4   | For X12 997 Version 2 Release 4 and lower                    |
| &DT_FA997V3R5   | For X12 997 Version 3 Release 5                              |
| &DT_FA997V3R7   | For X12 997 Version 3 Release 7                              |
| &DT_FA997V4R2   | For X12 997 Version 4 Release 2                              |
| &DT_FA997V4R6   | For X12 997 Version 4 Release 6 and higher                   |
| &DT_FA999V5R1   | Implementation Guide Syntax for X12 999 Version 5 Release 1  |
| &DT_FA999V3R3   | For UCS 999 Version 3 Release 3 and lower                    |
| &DT_FACONTRL    | For UN/EDIFACT earlier than Version 94B                      |
| &DT_FACONTRL94B | For UN/EDIFACT version 94B (Version 2, Release 1) and higher |
| &DT_FACONTRL94A | For UN/EDIFACT version 94A (Version 4, Release 1) and higher |
| &WDI_TA1_ACK    | For TA1 Acknowledgment                                       |
| &WDI_TDIENV_VAL | UNTDI TRADACOMS Envelope Validation Mapping                  |

**Note:** The Functional Acknowledgement maps must not be modified or deleted. If you need to make a customization, it is recommended that you copy the appropriate map make the required changes in the map copy.

The &DT\_FA997V3R7 Functional Acknowledgement map is used to generate functional acknowledgements for X12 version 3, release 7 and higher. The &DT\_FACONTRL94B Functional Acknowledgement map is used to generate functional acknowledgements for UN/EDIFACT version 2, release 1 (D94B) and higher. All of the Functional Acknowledgement maps listed must not be modified or deleted. If you need to make a customization, it is recommended that the appropriate map is copied and the needed change is made in the copied map.

**Note:** The &WDL\_TA1\_ACK map name cannot change. It is recommended that a **MapSwitch()** command be added to execute a custom TA1 map.

The source document definition for Functional Acknowledgment maps always uses Dictionary &FUNC\_ACK\_METADATA\_DICTIONARY and Document &FUNC\_ACK\_META. This is the data format definition for the functional acknowledgement records created during syntax validation of EDI data. The target document definition is a selection of EDI standard transaction definitions and are provided as follows:

*Table 21. Functional acknowledgement dictionaries*

| Dictionary | Description                                                 |
|------------|-------------------------------------------------------------|
| &DT99724   | For X12 997 Version 2 Release 4 and lower                   |
| &DT99735   | For X12 997 Version 3 Release 5 and lower                   |
| &DT99737   | For X12 997 Version 3 Release 7 and lower                   |
| &DT99742   | For X12 997 Version 4 Release 2 and later                   |
| &DT99746   | For X12 997 Version 4 Release 6 and later                   |
| &DT99951   | For X12 999 Version 5 Release 1 and later                   |
| &DT99933   | For UCS 999 Version 3 Release 3 and lower                   |
| &DTCTL     | For UN/EDIFACT earlier than Version 94B                     |
| &DTCTL41   | For UN/EDIFACT version 99A (Version 4 ) and later           |
| &DTCTL21   | For UN/EDIFACT version 94B (Version 2, Release 1) and later |

---

## Starting the Functional Acknowledgement map editor

The Functional Acknowledgement map editor opens when you select a map from the Mapping list window, as follows.

1. Click Mapping on the WebSphere Data Interchange Client Navigator bar.  
The Mapping Functional area opens, showing lists of existing maps.  
This window is used to access the list windows for each component type that occurs in the Mapping Functional Area.
2. Select the **Functional Acknowledgement Maps** tab to view the Functional Acknowledgement Map list window. The list window shows all of the Functional Acknowledgement Maps selected by the current query. Each row in a list window is opens information about a different Functional Acknowledgement Map.  
To display additional columns, click the scroll bar on the bottom of the page to scroll to the right or left. To alter the columns that display on the page, or to change which query is executed to produce the list, click the Modify Window Properties icon on the WebSphere Data Interchange Client Navigator bar. To create new queries, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.
3. To view a map or to add or change its information, double-click the row of the map you want to work with.  
The Functional Acknowledgement map editor opens, with the **Details** tab in front. You add information or make changes to maps through its tabs and related windows, as described in the following sections.

---

### Using the Functional Acknowledgement map editor

Using the Functional Acknowledgement map editor, you can show how data is to be translated from the source document to the target document. The editor works by displaying the source document definition on one side of the page and target document definition on the other. With the Functional Acknowledgement map editor you can also associate components of your source document with components of the target document by dragging source components and dropping them into the correct locations in the target document definition. Components include simple elements, such as fields in a data format or data elements in an EDI standard transaction, and compound elements, such as loops and record in a data format and segments in an EDI standard transaction.

You can map the simple elements in the source document definition to the simple elements in the target document definition in any of these patterns:

- One simple element to one simple element
- Several simple elements to one simple element
- One simple element to several simple elements

The Functional Acknowledgement map editor contains three tabs:

- The **General** tab contains the fields for you to enter and change map properties.
- The **Details** tab contains the fields for you to create and maintain the mapping commands.
- The **Comments** tab contains a field for you to type any comments you wish about the selected map.

The following procedures detail how to create maps. For information about viewing, copying, editing, renaming, deleting, and printing maps, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01. For information about exporting maps, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

---

### Creating a Functional Acknowledgement map

It is recommended that you do not create or modify Functional Acknowledgement maps. However, if you require a custom functional acknowledgement you can you the following procedure.

**Note:** To create a Functional Acknowledgement map, you need to have a functional acknowledgement document definition. Functional acknowledgement document definitions are:

- 997 and 999 in the X12 EDI Standard
  - CONTRL transaction in the EDIFACT EDI standard
1. In the Mapping list window, click the Functional Acknowledgement Maps tab.
  2. Click New on the list window tool bar.

The Create a Functional Acknowledgement Map window opens.

## Creating a Functional Acknowledgement map

3. Type a map name in the Map Name field.  
You can use both letters and numbers to identify your map. Letters display in capitals. You cannot type spaces within the name. A Functional Acknowledgement Map must have a unique map name. Click Show Existing Map Names to view a list of map names.  
Click Next to continue.
4. Enter a detailed description of the map in the Description field. This field is optional.  
You can enter the later using the **General** tab of the Functional Acknowledgement map editor.
5. The **Source or Target** opens. Use this page to define whether your map is source based, or target based.  
Select Source or Target by selecting the appropriate radio button.  
Click Next to continue.
6. The **Select Target** opens. Available target document definitions are displayed in the page's list box.  
Choose the functional acknowledgement to be used as the target document definition in your map.  
Click Next to continue.
7. The Confirmation window opens. Confirm the selections displayed. If they are correct, click Finish to save the information.  
After finishing the map information, the Functional Acknowledgement Map editor opens with the **Details** tab in front.

---

## Source document definition record layout

The Source document definition record layout is:

Interchange response Record - ID = INTHREC

Functional Group Response Loop:

Functional Group Response Record - ID = GRPHREC

Message Response Loop:

Message response Record - ID = MSGHREC

Implementation Reference (ST03 Value)

Segment Data Loop:

Data Segment Note Record - ID = SEGREC

Segment Context Record - ID = SEGCTX

Element Data Loop:

Data Element Note Record - ID = ELEREC

Element Context Record - ID = ELECTX

Message Response Trailer Record - ID MSGTREC

Functional Group Response Trailer Record - ID = GRPTREC

Source document definition record definitions &FUNC\_ACK\_META

Interchange response Record - ID = INTHREC

Interchange control reference

Sender Identification

Partner Identification code

Address for reverse routing

Recipient Identification

Partner Identification code

Routing address

Action, coded

Syntax error, coded

Segment tag, coded

Erroneous data element position

Erroneous component data element

Functional Group Response Record - ID = GRPHREC

Functional Identifier Code

Group Control Number

Application sender's identification

Partner Identification code

Application recipient's identification

Partner Identification code

Action, coded

Syntax error, coded

Segment tag, coded

Erroneous data element position

Erroneous component data element

Message response Record - ID = MSGHREC

Message reference number

Message type identifier

Common Access Reference

Message type version number

Message type release number

Controlling agency

Association assigned code

Action, coded

Syntax error, coded

Segment tag, coded

Erroneous data element position

Erroneous component data element

Implementation Reference (ST03 Value)

Data Segment Note Record - ID = SEGREC

Segment ID Code

Segment Position in Transaction Set

Loop Identifier Code

Segment Syntax Error Code

Segment Context Record - ID = SEGCTX

Text Char (35)

Reference Char (35)

## Creating a Functional Acknowledgement map

Data Element Note Record - ID = ELEREC  
Element Position in Segment  
Component Data Element Position in Composite  
Data Element Reference Number  
Data Element Syntax Error Code  
Copy of Bad Data Element

Element Context Record – ID = ELECTX  
Text Char (35)  
Reference Char (35)

Message Response Trailer Record - ID MSGTREC  
Message Acknowledgment Code  
Message Syntax Error Code  
Message Syntax Error Code  
Message Syntax Error Code  
Message Syntax Error Code  
Message Syntax Error Code

Functional Group Response Trailer Record - ID = GRPTREC  
Functional Group Acknowledge Code  
Number of Messages Included  
Number of Received Messages  
Number of Accepted Messages  
Functional Group Syntax Error Code  
Functional Group Syntax Error Code  
Functional Group Syntax Error Code  
Functional Group Syntax Error Code  
Functional Group Syntax Error Code

---

## Using Functional Acknowledgement Maps

You can work with and use Functional Acknowledgement maps in much the same way as you can with Data Transformation Maps. Many of the concepts explained for Data Transformation maps in Chapter 9, “Data Transformation mapping,” on page 121 also apply to Functional Acknowledgement maps. Note the following:

- Functional Acknowledgment maps always use the same data format for the source document. The dictionary name is &FUNC\_ACK\_METADATA\_DICTIONARY, and the data format name is &FUNC\_ACK\_META.

For more information about Functional Acknowledgement Mapping techniques and concepts, see the following sections:

- “Specifying qualification” on page 103
- “Applying map rules” on page 125
- “Compiling control strings” on page 116

WebSphere Data Interchange techniques for using literal keywords and other advanced mapping techniques are documented in Chapter 12, “Data Transformation mapping commands and functions,” on page 141.

---

## Chapter 12. Data Transformation mapping commands and functions

In addition to mapping a source element to a target element by using drag-and-drop, the Data Transformation Map editor includes a powerful set of commands. Using these commands, you can use conditional logic such as if/then/else, qualify repeating elements, save values in variables, create complex expressions, and many other functions. This appendix describes the command language used in the Data Transformation Map editor to perform these tasks.

**Note:** Although at the time of release this document contains a complete list of Data Transformation commands, upgrade and maintenance releases can add or change commands. See the WebSphere Data Interchange Client help for an updated list of commands.

When you create a map, you are basically creating a sequence of commands. Each command is attached to an object in the source or target document, such as a segment, data element, field, XML element, and so on. Whenever WebSphere Data Interchange finds an object in the source or target document, it executes any commands associated with it. You can also add commands before and after objects in the document. The WebSphere Data Interchange executes these commands, even if the elements before and after are not found. You can create the commands by dragging and dropping a source element to a target element (or a variable) or vice versa. This results in a **MapTo** (for source-based mapping) or **ForEach** (for target-based mapping) command or an assignment command being created. You can also create commands by right-clicking a node in the map command display to start the mapping command wizard.

The following sections describe the supported commands and functions, and provides information about other important features of the mapping language.

---

### Map variables

Map variables are used like variables in any programming language. They are an integral part of the WebSphere Data Interchange mapping command language. Variables are used to hold and manipulate values assigned to them by the user. WebSphere Data Interchange supports three types of variables: *local*, *global*, and *special variables*.

Local variables are unique to the map they are defined in. A local variable must be defined to a map before it can be used in that map. Local variables have a scope of *document* or *loop*. During translation, local variables defined with a scope of *document* are created at the start of every document and deleted at the end of the document. Variables defined with a scope of *loop* are created and initialized whenever a new loop iteration is started, and destroyed at the end of each loop iteration. A loop variable does not disturb the value of another variable with the same name at another level of looping. Local variables are maintained within the map in which they are defined. You can add, delete, and alter the properties of any local variable.

## Map variables

Special variables are a group of predefined variables used by WebSphere Data Interchange. They function much like local or global variables, except they each have a special purpose. A user can view properties of a special variable, but no changes can be made. Special variables always start with “DI”, which is reserved.

Special variables are write-only. That means you can set the value of a special variable, but cannot read it or use it in an expression.

Table 22 lists the special variables supported:

Table 22. Supported special variables

| Name      | Scope    | Type      | Length | Description                                                                                                                                                                  |
|-----------|----------|-----------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DIOutFile | Document | Character | 8      | Specifies the name of the file to which the translated data is written. It overrides any value that was specified in the Data Transformation rule or data format definition. |
| DIOutType | Document | Character | 2      | Specifies the file type to which the translated data is written. It overrides any value that was specified in the Data Transformation rule or data format definition.        |

## Naming variables

Variable names can be up to 30 characters long and can contain the characters 0–9, A–Z, a–z, and the special characters “@#\$\_”. Special variables always start with “DI”. Local variables cannot start with a number, dollar sign, or ampersand. Variables cannot be the same as any reserved word (command name, function name, or other reserved word).

Names can be in mixed case. Case is not used in determining uniqueness, for instance, the variable “MyValue” is the same as the variable “MYVALUE”.

**Note:** During translation, WebSphere Data Interchange might use variables whose names start with \$R to hold intermediate values. These variables are not visible in the map, but might appear in messages during translation.

## Literals

The Data Transformation Map editor uses the term *literals* differently from the send and Receive map editors. In the Data Transformation map editor, the term literal refers to a static value, such as a default value. The special literals used in the send and Receive maps are replaced by mapping commands in Data Transformation maps.

Literals can be either numeric or character strings. Character strings can be contained in either single or double quotation marks. For example:

"This is a character literal"



'This literal contains "quotes" within'

Numeric literals are made up of a sequence of digits, and can include a decimal point. (Negative numbers can also be used, but the negative sign is considered separate from the numeric literal.) For example:

99  
1.23

---

## Comments

You can enter comments by creating comment nodes in the mapping commands window. A comment group node adds several comments within a single node.

---

## Keywords

This section identifies keywords that have special meaning to WebSphere Data Interchange. In addition to the keywords listed in this section, all mapping commands, logical operators, comparison operators, and arithmetic operators are considered keywords.

**False** Used as a boolean value to test boolean expressions and set boolean variables.

**targetRoot**

Can be used on the **MapCall** command. See “Commands” on page 149 for more information.

**This** Can be used as a source or target path in mapping commands to indicate the current source or target element.

**True** Used as a boolean value to test boolean expressions and set boolean variables.

---

## Specifying a path

Paths are used to identify a source or target element in various mapping commands. To specify a source or target path in a command, drag the element from the source or target tree display to the command window.

The path indicates the position of a compound element or a simple element in the source or target document definition. Paths are used in various mapping commands to identify the compound element or simple element involved in the command. Paths are syntax dependent. For instance, all of the path examples use paths from EDI standard X12, version 3, release 6, transaction 837.

A path can include compound element identifiers and simple element identifiers. The path begins with a backslash to indicate that it starts at the root of the document. Each element of a path is separated by a backslash. The path is always terminated by two backslashes. An example of a path is:

```
\T 2\L 2300\S CLM 130\C C023 5\E 1332 2\\
```

## Specifying a path

This path translates to: table 2, loop Id 2300, CLM segment at position 130, composite data element C023 at position 5, and component data element 1332 at position 2.

The following path refers to the CLM segment at position 130 within table 2 and loop Id 2300:

```
\T 2\L 2300\S CLM 130\
```

**Note:** Source document elements are read-only. You cannot update values in the source document. Target document elements are write-only. You cannot read the value that has been assigned to an element in the target document.

---

## Forward and reverse references

WebSphere Data Interchange always has a *current position* in the source data. The current position is the path just located or obtained from the source data.

In source data, WebSphere Data Interchange is able to locate paths or simple elements before or after the current position. Doing this does not change the current position. The forward or reverse reference can be helpful in examining simple data or determining if a particular path exists.

For repeating items (loops, segments, elements, and so on), you cannot refer to a previous or later occurrence of the item. For example, if you are currently processing the second occurrence of a repeating segment, you cannot refer to an element from the first or third occurrence of the segment. If you have not yet started processing a repeating item, the first occurrence of the item is considered the current occurrence.

A more efficient option to using references is to save the information you need in variables. Then use the variable in place of the forward or reverse reference.

Forward and reverse references can be specified in expressions. A reference is specified by using a path.

---

## Data types supported by mapping commands and functions

WebSphere Data Interchange supports a wide variety of data types for simple elements. When the input data is parsed, the values are kept internally as one of the following data types:

|                  |                     |
|------------------|---------------------|
| <b>character</b> | Character data      |
| <b>int</b>       | Integer data        |
| <b>real</b>      | Floating-point data |
| <b>boolean</b>   | Boolean data        |
| <b>binary</b>    | Binary data         |

Variables can also be any of these data types. Literals are always either real or character. Implicit type conversion is done by the WebSphere Data Interchange when an element, variable, or literal is assigned to another type of element or variable, or is used in a function or command that expects a different type. Integer values can be converted to real or character; real values can be converted to integer or character;

character values can be converted to integer or real values. If an invalid type conversion is attempted, for example trying to assign a boolean variable to a character variable, an error is issued during the control string compilation.

Many of the commands and functions expect numeric values for their arguments or return a numeric value. All functions, commands, and expressions process numbers internally using data type real. However, because real values are implicitly converted to or from integers, integer variables and elements can be used in place of any of the real arguments or return values described in “Expressions.”

---

## Expressions

Expressions are commonly used in the mapping command language. The basic form of an expression is:

*token operator token [operator token [operator token [...]]]*

Where:

- |                 |                                                                                                                                                                                                                                                                                                            |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>token</b>    | <p>One of the following:</p> <ul style="list-style-type: none"> <li>• A variable, either local, global, or special</li> <li>• A path identifying a source simple element</li> <li>• A numeric constant, such as 1024</li> <li>• A string constant, such as 'ABCD'</li> <li>• A mapping function</li> </ul> |
| <b>operator</b> | <p>One of the following types of operators:</p> <ul style="list-style-type: none"> <li>• Logical</li> <li>• Comparison</li> <li>• Arithmetic</li> <li>• Unary</li> </ul>                                                                                                                                   |

Quotation marks must be used at the start and end of a string constant. Either single (') or double quotation marks (") can be used, but whichever is used to start the string constant must also be used to end it.

All variables, literals, and simple source and target elements have a specific data type. WebSphere Data Interchange converts the value to the appropriate type for the expression if needed. If it cannot convert the value to the appropriate type because the types are incompatible (for example, boolean to binary), an error is issued during the control string compilation. If it cannot convert the value to the appropriate type because the values are incompatible, (for example "abc" to real), WebSphere Data Interchange issues an error or warning during translation.

## Logical operators

Data Transformation maps use the following logical operators:

- **AND** returns a value of **True** if both conditions are true. **False** is returned when either condition is false.
- **OR** returns a value of **True** if either condition is true. **False** is returned when neither condition is true.

## Operators

- **NOT** reverses the boolean result of an expression. For logical expressions, **NOT(expression)** returns **False** if the result of **expression** is **True**. **NOT(expression)** returns **True** if **expression** is **False**.

**NOT** has a short form that can be specified as an exclamation point (!). For example, the following are valid:

```
NOT(expression)
!(expression)
NOT(myBoolFlag)
!myBoolFlag
```

**Note:** Case is not relevant with the logical operators. They can be entered as uppercase, lowercase, or mixed.

## Comparison operators

Comparison operators tell the WebSphere Data Interchange to compare two numeric objects. WebSphere Data Interchange processes comparison operators in the same way that it handles logical operators. If the comparison is true, **True** is returned. If the comparison is false, **False** is returned. Table 23 lists the comparison operators supported.

*Table 23. Comparison operators*

| Operator | Alternate | Description                                                  |
|----------|-----------|--------------------------------------------------------------|
| EQ       | =         | The first value is the same as the second                    |
| GE       | >=        | The first value is greater than or equal to the second value |
| GT       | >         | The first value is greater than the second value             |
| LE       | <=        | The first value is less than or equal to the second value    |
| LT       | <         | The first value is less than the second value                |
| NE       | != or <>  | The first value is not equal to the second value             |

Case is not relevant with the comparison operators. They can be entered as uppercase, lowercase, or mixed.

**Note:** The comparison operators are for numeric (integer and real) data types only. Character strings can be compared using the string comparison functions.

## Arithmetic operators

Data Transformation maps support addition, subtraction, multiplication and division of numeric values. A modulus operator is also supported. The modulus operator (%) will return the remainder of the first operand divided by the second.

Division by zero using the division operator ( / ) will produce an error. Division by zero using the modulus operator (%) will also produce an error.

## Unary operators

One unary (single component) operator is supported by the WebSphere Data Interchange, the dash (-). This operator changes the sign of the expression it precedes. For example, if the value of *var1* is 9, specifying *-var1* returns -9. It does not change the value of *var1*.

## Order of precedence

During processing, all expressions are evaluated from left to right. The order of precedence is:

1. Unary minus (-)
2. Multiple (\*), Divide (/)
3. Modulus (%)
4. Addition (+), Subtraction (-)
5. Relational Operators (GT, GE, LT, LE, EQ, NE)
6. Logical NOT Operator
7. Logical AND Operator
8. Logical OR Operator

Precedence can be overridden by using parentheses within an expression. For example,  $2+3*5$  equals 17 because the multiplication is done first, then the addition.  $(2+3)*5$  equals 25 because the parentheses indicate that the addition is done first, then the multiplication.

---

## Assignment

A value can be assigned to any variable or any target simple element. This is accomplished using an assignment statement. An assignment statement has the following format:

```
target = expression
```

Where:

**target** Is any defined variable or simple element in the target document definition.

**expression** Is any valid expression.

WebSphere Data Interchange attempts to convert the result from *expression* to the same data type of *target*, if needed. If it is unable to make the conversion, an error is issued.

---

## Conditional commands

### If / Elseif / Else / Endif

The **If** command is used to conditionally perform one or more mapping commands. The **If** command uses the following format:

## Commands

```
If (condition)
 mapping commands
[ElseIf (condition)
 mapping commands]
[ElseIf (condition)
 mapping commands]
[Else
 mapping commands]
EndIf
```

Where:

|                         |                                                                      |
|-------------------------|----------------------------------------------------------------------|
| <b>condition</b>        | Any valid expression that evaluates to <b>True</b> or <b>False</b> . |
| <b>mapping commands</b> | One or more mapping commands.                                        |

The **If** command marks the beginning of the If condition block. **EndIf** is used to mark the ending of the If condition block.

When the **If** command is encountered by the WebSphere Data Interchange, the *condition* is evaluated. When *condition* evaluates to **True**, the mapping commands immediately following the **If** command is executed. If *condition* evaluates to **False**, WebSphere Data Interchange looks for an **Elseif** statement within the If condition block. If an **Elseif** statement is found, its associated *condition* is evaluated. When *condition* evaluates to **True**, the mapping commands immediately following the **Elseif** statement is executed. If *condition* evaluates to **False**, WebSphere Data Interchange looks for the next **Elseif** statement within the If condition block. When all **Elseif** statements have been tested and evaluated to False, WebSphere Data Interchange looks for an **Else** statement within the If condition block. If an **Else** statement is found, the mapping commands immediately following the **Else** statement is executed.

The **Elseif** and **Else** statements are optional. When present, an **Else** statement always precedes the **EndIf** statement. **Elseif** statements always follow the If statement and precede the **Else** statement (if present) or the **EndIf** statement (when the **Else** statement is not present).

WebSphere Data Interchange examines the If and **Elseif** statements sequentially until an associated *condition* evaluates to **True**. When a *condition* has evaluated to **True**, only the mapping commands associated with the corresponding If or **Elseif** statement is executed. After the mapping commands have been executed, WebSphere Data Interchange exits the If condition block and resume processing after the **EndIf** statement. If no *condition* evaluates to **True**, the mappings commands associated with the **Else** statement is executed and WebSphere Data Interchange resumes processing after the **EndIf** statement. If no *condition* evaluates to **True** and there is no **Else** statement, no mapping commands within the If condition block is executed and the translator resumes processing after the **EndIf** statement.

---

## Commands

Mapping commands all perform a specific action as described in the following sections. The command name is not case sensitive. For example, the **Error** command can also be specified as **ERROR**. Most commands can take any expression for their arguments, as long as the expression evaluates to the appropriate type (or its result can be converted to the appropriate type). The only exception is when the argument is specified as a *sourcepath* or *targetpath*. For those commands, the argument must be a source or target path. A description of each of the supported commands is provided in the following sections.

### CloseOccurrence

Use the **CloseOccurrence** command to close the current occurrence of a repeating element and force the creation of another instance of the element. This command can be used for both source-based and target-based maps.

In target-based documents, the **CloseOccurrence** command prevents overlaying of data in the Target document. The command closes the current occurrence of a repeating target element and forces the creation of another instance of the element.

Use the CloseOccurrence command in the following instances:

- Forward and Reverse References  
References or mapping to elements not at the current position might need a **CloseOccurrence** command.
- Mapping between Elements  
Mapping not attached to a particular element might need a **CloseOccurrence** command.
- Qualification on non-repeating or non-compound elements.  
There is an Implied CloseOccurrence at the loop level. A **CloseOccurrence** might be needed to close the occurrence of the non-repeating target element before the implied CloseOccurrence is executed.

**Note:** For Multi-Occurrence Qualifications, there is an implied **CloseOccurrence** with each instance of the source or target path.

The **CloseOccurrence** command has the following format:

```
CloseOccurrence(targetpath)
```

Where:

**targetpath** Specifies a repeating target element that is "closed" to any new elements in the current occurrence. Any additional mappings to this element or its children result in a new occurrence of the *targetpath* element.

The **CloseOccurrence** command can be shortened to **CloseOccur**.

## Commands

### Example

Suppose your source document has the following structure:

```
Rec1
 Field1
 Field2
 Field3
 Field4
```

Suppose your target document has the following structure:

```
Seg (repeats)
 Seg01
 Seg02
```

Field1 and Field2 are to be mapped to the first occurrence of Seg (Seg01 and Seg02), and Field3 and Field4 to the second occurrence of Seg (again, Seg01 and Seg02). To accomplish this, you would:

- Map Field1 to Seg01, Field2 to Seg02.
- Insert a `CloseOccurrence(Seg)` command to close the first occurrence of Seg.
- Map Field3 to Seg01, Field4 to Seg02, which creates a second occurrence of Seg.

## Create

The **Create** command is used to create a specified compound element in the target data. The **Create** command has the following format:

```
Create(targetpath)
```

Where:

**targetpath** identifies the path in the target document definition that is to be created. *targetpath* must be a compound map node, such as an XML element. To create simple nodes such as an attribute or an element value, the assignment command or **MapFrom/MapTo** commands are used.

## Error

Use the **Error** command to issue an error condition. This command enables you to establish your own errors for a translation. Typically, the error is issued from within an If conditional block.

When the **Error** command is used in a validation map, the error message is written to the PRTFILE as the validation map is processed. Because the validation map is run as a pre-processing step before the normal EDI standard syntax validation, this means that all PRTFILE messages generated by the **Error** command appears before the TRxxxx messages generated by the EDI validation, and before the messages generated by the **FAError** commands.

The **Error** command has the following format:

```
Error(real level, real code, char text)
```



Where:

- level** Indicates the severity of the error. It is a value of 0, 1, or 2.
- code** Is the unique error code that is associated with the error. This can be any value from 5000 to 5999.
- text** Is the string value, which is included in an error message issued by WebSphere Data Interchange when this command is executed.

The message is issued as a UT0033 error message. Within this message, *text* and *code* are included.

The *level* that is assigned affects the extended return code from the translator, and thus the JCL condition code in the WebSphere Data Interchange utility. If *level* exceeds the acceptable error level specified in the document usage, the translation is not successful.

## ErrorContext

To create the implementation guide syntax (IGS) context, the **ErrorContext** mapping command can be used following the **FAError** or **FAErrorPath** mapping command in the validation map.

The **ErrorContext** command has the following format:

```
ErrorContext (char Text, char Reference)
```

Where:

**Text** Is the IGS Name. Maximum characters is 35.

**Reference**

Is the IGS Reference. Maximum characters is 35.

Example:

```
FAError(1, 5001, "7", "HL22\SBR02\Subscriber Individual Relationship Code Element Value in Error", "E
ErrorContext ("Required only when Subscriber is Patient", "NSF Reference: DA0-17.1")
```

The **ErrorContext** is associated with the previous **FAError** or **FAErrorPath**. A previous **FAError** or **FAErrorPath** for error type T (Transaction), G (Group), or I (Interchange) is ignored because the 999 does not support context reporting at these levels.

## FAError

Use the **FAError** command to set a functional acknowledgement code. You can only use this command in validation maps.

When an **FAError** command occurs in a validation map, the **FAError** information is held and associated with the current position in the source data. When the remaining validation is done, such as checking for missing mandatory elements and segments, the

## Commands

validation processing checks to see if there is any FAError information for this position. If so, the FAError information is written to the PRTFILE and added to the input for the Functional Acknowledgement map. Because the FAError information is held and associated with the current position, resulting PRTFILE messages are interspersed with any normal validation messages, such as missing mandatory segments and elements.

Use this command at any point at which you want to identify an error condition. This command enables you to establish your own error codes for translation. Typically, the error is issued from within an If conditional block.

The **FAError** command has the following format:

```
FAError(level, code, facode, msgtext)
```

Where:

|                |                                                                                                                                                                                                                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>level</b>   | Indicates the severity of the error. It is a value of 0, 1, or 2                                                                                                                                                                                                                                        |
| <b>code</b>    | Is the unique error code that is associated with the error. This can be any expression that evaluates to an integer with a value from 5000 to 5999.                                                                                                                                                     |
| <b>facode</b>  | Is an acknowledgement code that is associated with this error. This value is placed in the functional acknowledgement if a functional acknowledgement is being generated. This can be any expression that evaluates to a string. You must ensure that the string is valid for the specific syntax type. |
| <b>msgtext</b> | Is any expression that evaluates to a string. It is included in an error message issued by WebSphere Data Interchange when this command is executed.                                                                                                                                                    |

The message is issued as a UT0033 error message. Within this message, *text* and *code* are included.

The *level* that is assigned affects the extended return code from the translator, and thus the JCL condition code in the WebSphere Data Interchange utility. If *level* exceeds the acceptable error level specified in the document usage, the translation is successful.

The **FAError** command has been extended to with some new optional parameters. The new parameters enable you to specify additional type/position information instead of just basing this information about the segment, element, or subelement where it was issued. The new command takes the following parameters:

```
FAError(level, code, faCode, msgText, errType, segId, eltPos, subEltPos)
```

The *level*, *code*, *faCode*, and *msgText* parameters are the same as before. The new parameters are as follows:

### **errType (optional, data type character)**

- I** Indicates an interchange level error (such as bad interchange header/trailer).
- G** Indicates a group level error (such as bad group header/trailer).

- T** Indicates a transaction level error (such as bad transaction header/trailer).
- S** Indicates a segment error (such as for missing segment).
- E** Indicates a data element error (such as for missing element or subelement).

If this is not specified, the default is determined from the parent node. If the parent node is a simple or composite data element, errType E is assumed. Otherwise, errType S is assumed.

**Note:** If this parameter is used, it must be entered as string literal in the command (such as I, G, T, S, or E.). Variables and other expressions are not permitted so that the map editor can check for valid values, instead of waiting for invalid values to cause a runtime error.

### segId

This is the segment id, for example "CUR", that is to be returned in the functional acknowledgement (if applicable). If this parameter is not specified or if an empty string ("") is used, then the segment id is determined as follows:

- If the **FAError** command is issued from within a segment (including within a data element or subelement), the id of that segment is used.
- Otherwise, the segment id of the next segment found in the loop or transaction is used.

Default values are not used for (I)nterchange, (G)roup, or (T)ransaction type errors.

### eltPos (optional, data type integer)

This is the element position within the segment. For example the second element of segment would have a value of "2". This value is also returned in the functional acknowledgement if applicable. If this parameter is not specified or is 0, then the element position is determined as follows:

- If the **FAError** command is issued from within a simple or composite data element (including within a subelement), the position of that element is used.
- Otherwise, the element position is omitted.

**Note:** This parameter is ignored for segment errors. Default values are not used for (I)nterchange, (G)roup, or (T)ransaction type errors.

### subeltPos (optional, data type integer)

This is the subelement position within the composite data element. For example the third subelement of a composite would have a value of "3". This value is also returned in the functional acknowledgement if applicable. If this parameter is not specified or is 0, then the subelement position is determined as follows:

- If the **FAError** command is issued from within a subelement, the position of that subelement is used.
- Otherwise, the element position is omitted.

## Commands

**Note:** This parameter is ignored for segment errors. Default values are not used for (I)nterchange, (G)roup, or (T)ransaction type errors.

If FAError is issued with errType="I", this flags the entire interchange as being invalid. This is only recognized if it is the first transaction in the interchange, and is only used for input validation. If it is not the first transaction in the interchange or the map is used for output validation, then this FAError is ignored. Segment and data element errors within the transaction are reported, and validation and processing of the remaining transactions in the interchange is skipped. The segment/element/subelement is taken from the optional parameters as described.

If FAError is issued with errType="G", this flags the entire group as being invalid. This is only recognized if it is the first transaction in the group, and is only used for input validation. If it is not the first transaction in the group, groups are not being used, or the map is used for output validation, then this FAError is ignored. Segment and data element errors within the transaction are not reported, and validation and processing of the remaining transactions in the group is skipped. The segment/element/subelement is taken from the optional parameters as described.

If FAError is issued with errType="T", this flags the entire transaction as being invalid. It is only used for input validation. If the map is used for output validation, then this FAError are ignored. Segment and data element errors within the transaction are not reported. The segment/element/subelement is taken from the optional parameters as described.

If FAError is issued with errType="S", this causes a segment error record to be generated after the other segments in the loop or transaction are processed. The segment information is set as follows:

- The segment id (AK301 for X12, N/A for EDIFACT) is set from the segld value, or calculated as described.
- The segment position (AK302 for X12, UCS01 for EDIFACT) is set based on the parent.
  - If the **FAError** command is on the root node or within a LOOP node, then the segment position sets the next segment found in the transaction.
  - If the **FAError** command is within a segment node (including within data elements or subelements), then the current segment position is used. This would be similar to the **FAError** command, but the segld value overrides the segment id if specified.

The element or subelement information is not used.

If the FAError is issued with errType="E", this causes a data element error record to be generated after the other data elements or subelements are processed. The segment position (AK3 or UCS) is taken from the current segment information, so errType="E" can only be used within a segment or data element. If the FAError with errType="E" is used outside of a segment (that is, between segments), an error is issued when the map is compiled. The element and subelement information is set from the eltPos and subeltPos as described.

## FAErrorPath

The **FAErrorPath** command has the following format:

```
FAErrorPath(level, code, facode, msgtext, errType, sourcepath)
```

Where:

**level** Indicates the severity of the error. It is a value of 0, 1, or 2 .

**code** Is the unique error code that is associated with the error. This can be any expression that evaluates to an integer with a value from 5000 to 5999.

**facode**

Is an acknowledgement code that is associated with this error. This value is placed in the functional acknowledgement if a functional acknowledgement is being generated. This can be any expression that evaluates to a string. You must ensure that the string is valid for the specific syntax type.

**msgtext**

Is any expression that evaluates to a string. It is included in an error message issued by WebSphere Data Interchange when this command is executed.

**errType (data type character)**

- I** Indicates an interchange level error (for example, bad interchange header/trailer).
- G** Indicates a group level error (for example, bad group header/trailer).
- T** Indicates a transaction level error (for example, bad transaction header/trailer).
- S** Indicates a segment error (for example, for missing segment).
- E** Indicates a data element error (for example, for missing element or subelement).

**sourcepath**

This is used to identify the segment or element position where the **FAErrorPath** command is issued. If the SourcePath is not found, the current position is used.

## ForEach

Use the **ForEach** command to execute mapping commands for each occurrence of the specified source node. Each occurrence of the source results in a new occurrence of the current target element.

You can include multiple **ForEach** command blocks within a single target element. The target element can be either a repeating node or a non-repeating node, and can be either a simple or a compound node.

If the target node is not repeating, you might need to use the **Qualify** command or conditional logic (If/Then/Else) so that only one occurrence of the source value is written to the target. If the target is not repeating and multiple values are written to it, later values overwrite the earlier values.

## Commands

You can only use the **ForEach** command in a target-based map.

The **ForEach** command has the following format:

```
ForEach(sourcepath)
```

Where:

### **sourcepath**

Identifies an element in the source document definition. Each occurrence of the element results in the creation of the target element which this command is under.

Establishes a current source position for the commands within the **ForEach** block.

The mapping commands within each **ForEach** command are executed for each occurrence of the specified source node (contrast this with the **Qualify** command, in which the mapping commands are executed only if a specific expression evaluates to True). See “Example 1.”

Forward and backward references to source elements outside the domain are performed with the same limitations that they have in source-based mappings. That is, you can refer to other siblings, cousins, and so on, but cannot refer to other elements within other occurrences of the *sourcepath*. Also, any nested ForEach elements must be within the current domain. This is illustrated in “Example 2” on page 157.

A single **ForEach** command can apply to multiple subdomains. “Example 3” on page 157 provides an example of this.

### **Example 1**

A map might be represented something like:

```
N1 Loop (element node)
 ForEach (\Order\Header\Sender\\) (command node)
 N1 (element node, might contain commands)
 N2 (element node, might contain commands)
 ...
 ForEach (\Order\Header\Receiver\\) (command node)
 N1 (element node, might contain commands)
 N2 (element node, might contain commands)
 ...
```

In this example, there are two **ForEach** commands within the N1 Loop. Each occurrence of source elements Sender or Receiver results in a new occurrence of a target N1 Loop. When a source Sender element is being processed, the mapping commands within the first **ForEach** block are executed to create the target N1 Loop occurrence. When a source Receiver element is processed, the mapping commands within the second block are executed

## Example 2

A source document might have the following structure (where "\*" indicates repeating element):

```
Order
 Header
 PONumber
 Date
 NameAddress*
 Type
 Name
 ...
 DetailLoop*
 LineItemNumber
 SubDetailLoop*
 Description
 ...
 Trailer
 Total
```

If your map has a **ForEach(\Order\DetailLoop\)** command, the domain for this block is the current DetailLoop occurrence. Commands in this block can refer to nonrepeating elements outside of the domain, such as `\Order\Header\PONumber\` and `\Order\Trailer\Total\`. The commands can also refer to repeating elements outside the domain, such as `\Order\Header\NameAddress\`. However, with forward and backward references to repeating elements it is more difficult to predict which occurrence is used (it depends on what commands have been processed previously). If another **ForEach** command is nested within the `ForEach(\Order\DetailLoop\)`, it must refer to an element within `\Order\DetailLoop\`, such as `\Order\DetailLoop\SubDetailLoop\`. The nested **ForEach** command cannot refer to an element outside the current domain, such as `\Order\Header\NameAddress\`.

## Example 3

For example, a source document has the following structure:

```
Order
 DetailLoop
 SubDetailLoop
 SubDetailLoop
 DetailLoop
 SubDetailLoop
 SubDetailLoop
```

If the current domain is the root of the document (Order), and you specify `ForEach(\Order\DetailLoop\SubDetailLoop\)` without the intermediate `ForEach(\Order\DetailLoop\)`, all four occurrences of SubDetailLoop (across both DetailLoop occurrences) are processed. If only the SubDetailLoop occurrences within a current DetailLoop are to be used, a **ForEach(\Order\DetailLoop\)** command is specified, then a **ForEach(\Order\DetailLoop\SubDetailLoop\)** command is specified within it.

## Commands

### HLLevel

Use the **HLLevel** command to create an HL loop within an EDI source or target based map.

The **HLLevel** command has the following format:

```
HLLevel(char levelcode)
```

Where:

**levelcode** Indicates the level code for this level. You must select this value from the list of level codes that are valid for the code list for HL03 (element 735). This value identifies the context of a series of segments following the current HL up to the next occurrence of an HL, or the end of the loop.

If you are working with a source-based map:

- Drag a structure or record from the target window and drop it on to the HL loop in the source window. A **MapTo** command is created under the loop.
- To create the underlying HL nesting levels or children for this HL level, right-click the HL loop and select **AddChild**, or select **AddPeer** to create a sibling.

If you are working with a target-based map:

- Right-click the HL loop again and select the **ForEach** command
- Drag the source path to the command wizard window.
- To create the underlying HL nesting levels or children for this HL Level, right-click the HL loop and select **AddChild**, or select **AddPeer** to create a sibling.

For more information about HL loops, see Appendix B, “Hierarchical loops,” on page 297.

### MapCall

Use the **MapCall** command to indicate that a new map must be used to process the data within the current source element (for source-based maps) or the specified source element (for target-based maps).

When this command is encountered, a new copy of the translator is loaded. It receives the data from the current element to use as its input document. The element can be a simple element (for example, the BIN02 element in a BIN segment) or it can be a compound element (a subtree or subset). The copied translator shares global variables with the parent translator.

When the copied translator has completed translation, it terminates, and control is returned to the parent translator. The parent translator checks if the imbedded translation was successful, and if so, resumes its own translation processing.

The output depends on the specified targetPath and the target document definition in the called map.



The `MapCall` command has the following format:

```
MapCall(sourcepath, char mapname, targetpath)
```

Where:

- sourcepath** Identifies a source element that is to be used as the root of the source tree for the imbedded map. If the *sourcepath* element does not exist in the input message, the `MapCall` command is not executed.
- For source-based maps, this value must be the keyword *This*, which indicates the current source element.
  - For target-based maps, the *sourcepath* element must be within the current domain.
- mapname** Is a character string that specifies the map name. You can specify a literal value or a more complex expression that evaluates to a character string.
- targetpath** Identifies the target element where the output of the imbedded map goes.
- If this is a simple binary element, such as a `BIN02`, the output from the called map is serialized and placed into this element. Use of compound elements and other data types for the `targetPath` on the `MapCall` command is not supported in Version 3 Release 2.
  - If this is the keyword *targetRoot*, the following logic is applied:
    - If the target document on the submap is the same as the target document on the primary map, the submap output is inserted into the same message as the primary map output. Children of the root in the submap output become children of the root in the primary map output.
    - If the target document on the submap is different than the target on the primary map, a separate message output is created.

## MapChain

Use the **MapChain** command to indicate that the document needs to be translated by another map after the current translation has completed. The subsequent translation only occurs if the current translation is successful.

The **MapChain** command has the following format:

```
MapChain(char mapname)
```

Each **MapChain** command encountered during translation of a map indicates that the document is translated an additional time using the newly specified map. This process stops if an error is encountered in any one translation. The output from a translation is independent from the other translations.

## MapFrom

You can use the `MapFrom` command in the following ways:

## Commands

- To move data from a repeating simple element in the target document definition to a corresponding repeating simple element in the source document definition.  
When you use the `MapFrom` command on a repeating target node, multi-occurrence mapping is required, and the `MapFrom` command must be included within a `ForEach` command.
- To move data from a nonrepeating simple element or variable in the target document definition to a corresponding simple element in the source document definition.

You cannot use the `MapFrom` command with compound elements.

Use the `MapFrom` command for target-based maps only: you must use the `MapTo` command for source-based maps. If the map is source-based and this command is used, the compiler command processor issues an error.

The `MapFrom` command has the following format:

```
MapFrom(expression)
```

Where:

**expression** Is the expression whose value is placed into the simple element. This expression can be any data type, as long as it is compatible with the target element that it is being mapped to.

The expression is evaluated, just as it would be for source-based transformation. The expression can be a simple source element, a variable, or a complex expression. If the expression does not evaluate to an empty string, the expression result is saved in the current target node. This is functionally equivalent to:

```
If (not IsEmpty(TrimRight(expr)))
 This = expr
```

## MapSwitch

Use the **MapSwitch** command to indicate that the document needs to be translated by another map instead of the current map. Any translation performed by the current map is terminated and the document is translated by the map specified in the **MapSwitch** command.

**MapSwitch** has the following format:

```
MapSwitch(char mapname)
```

Use this command when data in the document must be inspected before it can be determined which map to use. With this command, you can switch the map dynamically based on the data that is contained in the document. You can create a map that initially examines the data in the document. Only the compound and simple elements necessary to make a mapping decision are mapped. The map that is used to translate the document is determined based on the mapped elements. Then use conditional mapping commands and the **MapSwitch** command to begin translating the document with the correct map.

## MapTo

You can use the `MapTo` command in several different ways:

- To associate a repeating compound element in the source document definition to a corresponding repeating compound element in the target document definition.
- To move data from a repeating simple element in the source document definition to a corresponding repeating simple element in the target document definition.
- To move data from a nonrepeating simple element or variable in the source document definition to a corresponding simple element in the target document definition.

A `MapTo` command is generated when you drag a source element to a target element or a target element to a source element.

Use the `MapTo` command for source-based maps only: you must use the `MapFrom` command for simple assignments for target-based maps, or the `ForEach` command for multi-occurrence mapping with target-based maps. If the map is target-based and the `MapTo` command is used, the compiler command processor issues an error.

The `MapTo` command has the following format:

```
MapTo(targetpath [, expression])
```

Where:

- |                   |                                                                                                                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>targetpath</b> | Identifies the path in the target document definition that is being mapped.                                                                                                                                   |
| <b>expression</b> | Is the expression whose value is placed into the simple element identified by <i>targetpath</i> . This expression can be any data type that is compatible with the target element that it is being mapped to. |

When placed on a repeating or nonrepeating compound element and only *targetpath* is specified, the compound element in the source document definition is associated with the compound element identified by *targetpath* in the target document definition. Each occurrence of the source compound element results in the creation of a corresponding target compound element.

When placed on a repeating simple element and only *targetpath* is specified, the simple element in the source document definition is associated with the simple element identified by *targetpath* in the target document definition. Each occurrence of the source simple element results in the creation of a corresponding target simple element and data being moved from the source simple element to the target simple element. This is equivalent to the assignment statement:

```
targetpath = (current source element)
```

When placed on a nonrepeating simple element and only *targetpath* is specified, the simple element in the source document definition is associated with the simple element identified by *targetpath* in the target document definition. Encountering the source

## Commands

simple element results in the creation of a corresponding target simple element and data being moved from the source simple element to the target simple element. This is equivalent to the assignment statement:

```
targetpath = (current source element)
```

When *targetpath* is specified with *expression*, the result of *expression* is moved to the simple element identified by *targetpath* in the target document definition. This is equivalent to the assignment statement:

```
targetpath = expression
```

## Qualify and Default

The **Qualify** command is used on repeating compound or repeating simple elements in the source document definition. It is used to indicate that a specific iteration or iterations of the elements are to be handled differently than other iterations of the element. For instance, you might want to say that the first iteration of a loop is handled differently than all other iterations of the loop. The **Qualify** command uses the following format:

```
Qualify(bool boolExpr)
```

Where:

**boolExpr** Is the boolean value or expression that, when True, indicates the mapping commands within the qualification is executed by the translator.

*boolExpr* can indicate if this is a particular occurrence of a repeating compound or simple element. Values of variables or simple elements in the source document definition can also be checked to determine when to execute the qualification. For instance, you can check to see if a specific simple element has a value of "ABC". If it does, the mapping commands within the qualification is executed by the translator.

### Example 1

```
Qualify (StrComp(\Table 1\310 0 N1 Loop\310 0 N1\1 M 98\\, "ZZ") = 0)
```

In this example, the value contained in simple element "\Table 1\310 0 N1 Loop\310 0 N1\1 M 98\\" is compared to the literal "ZZ". If the function returns 0 (values are equal), the mapping commands within the qualification are executed.

### Example 2

```
Qualify(Occurrence() = 2 OR Occurrence() = 3)
```

In this example, the mapping commands within the qualification are executed if it is the second or third occurrence of the simple or compound element.

### Example 3

```
Qualify(Occurrence() =
2 AND StrComp(\Table 1\310 0 N1 Loop\310 0 N1\1 M 98\\, "ZZ") = 0)
```

In this example, the mapping commands within the qualification are executed if it is the second occurrence of the current compound source element and simple element "\Table 1\310 O N1 Loop\310 O N1\1 M 98\\" is "ZZ".

The Occurrence and StrComp functions are described in more detail in "Functions" on page 175. Any boolean expression or function can be used as the expression on the **Qualify** command.

The **Default** command can be used to specify the commands that are executed if none of the Qualify expressions evaluate to True.

## SetElementAttribute

This command controls how leading zeros, leading and trailing blanks, left and right justification, and empty elements are handled when they are output for an EDI optional Data Element. This command does not affect the values of simple elements during translation. It only affects the formatting of simple elements when they are written to the output document. This command can be specified anywhere in a Data Transformation Map. The SetElementAttribute command has the following format:

```
SetElementAttribute (targetPath, attributeName, attributeValue)
```

Where:

### targetPath

A path in the target document definition that identifies a simple or compound element. The reserved word TargetRoot can be specified to indicate that the attribute value is the default for the element attribute for all simple elements within the document. The keyword **This** can be used to indicate the command is being applied to the parent element.

### attributeName

The name of the element attribute you want to set. See "Element attributes" for additional information about element attributes and a complete list of attributes that you can set. This parameter is typically a string constant, but can be a variable, a path identifying a simple element in the source document definition, or an expression that evaluates to a character string.

### attributeValue

The value for the element attribute. This can be a numeric or string constant, variable, path identifying a simple element in the source document definition, or an expression.

**Note:** Excessive use of the SetElementAttribute command increases memory usage during map execution.

## Element attributes

The following sections describe each of the element attributes.

**TrimLeadingZeros:** Indicates how to treat leading zeros for values of simple elements declared to be of a numeric type. Valid values are:

**Y** Specifies that leading zeros beyond the ones position from the beginning of a

## Commands

simple element value up to the minimum length of the simple element are trimmed before adding the value to the output document. At least one zero is produced before the decimal point for numeric data types that contain a decimal point, such as real (R). At least one zero is produced before the decimal digits for numeric data types that have an implied decimal, such as numeric (N).

The minimum length of the simple element is not considered when the NumValueTooShort element attribute has been specified and the TrimLeadingZeros element attribute has been specified with a value of Y. Leading zeros are always removed to the first significant non-zero digit to the left of the decimal position.

- N** Do not trim leading zeros at the beginning of a simple element value before adding it to the output document.

Formatting notes for each document syntax type:

**XML** The default setting for this element attribute is N (do not remove leading zeros). All XML values have a minimum length 1. A value is considered to be numeric if the value contains all numbers or numbers with a decimal. The evaluation to determine if an XML value is numeric is performed after any optional operations to remove leading or trailing blanks. By default, trailing blanks are automatically removed for XML values while leading blanks are not automatically removed.

### EDI Standard

The default setting for this element attribute is Y (remove leading zeros). The minimum length is specified in the definition of the Data Element. A Data Element is considered to be numeric if its data type is real or numeric.

### Data Format

The default setting for this element attribute is N (do not remove leading zeros). The minimum length for a fixed length Data Format Field is always the same as the defined maximum length for the field. The minimum length of a field in a comma delimited Data Format is one unless dictated otherwise by the data type. For instance, a Field with a data type of real with two significant decimal positions (R2) has a minimum length of 4 (as in the value 0.00). A Data Format Field is considered to be numeric if its data type is real, numeric, integer, binary, hexadecimal, packed decimal, or zoned decimal.

**TrimTrailingZeros:** Indicates how to treat trailing zeros to the right of the decimal point for values of simple elements declared to be of a numeric type. Valid values are:

- Y** Trims trailing zeros to the right of the decimal point beginning at the end of the value and continuing left until a non-zero decimal digit or the minimum number of decimal digits is reached before adding it to the output document.
- N** Does not trim trailing zeros at the end of a simple element value before adding it to the output document.

Formatting notes for each document syntax type:

**XML** The default setting for this element attribute is N (do not remove trailing zeros).

Zero decimal digits are required for all XML values. A value is considered to be numeric if the value contains all numbers or numbers with a decimal. The evaluation to determine if an XML value is numeric is performed after any optional operations to remove leading or trailing blanks. By default, trailing blanks are automatically removed for XML values while leading blanks are not automatically removed.

#### **EDI Standard**

The default setting for this element attribute is Y (remove trailing zeros). The number of required decimal digits is specified using the data type in the definition of the Data Element. A Data Element is considered to be numeric if its data type is real or numeric.

#### **Data Format**

The default setting for this element attribute is Y (remove trailing zeros). The number of required decimal digits is specified using the data type in the definition of the Field. A Data Format Field is considered to be numeric if its data type is real, numeric, integer, binary, hexadecimal, packed decimal, or zoned decimal.

**TrimLeadingBlanks:** Indicates how to treat leading white space at the beginning of a simple element value. White space includes blanks, carriage returns, line feeds and tab characters. Valid values are:

- Y** Removes all leading white space from the beginning of a simple element value up to the minimum length of the simple element before adding it to the output document. The minimum length of the simple element is not considered when the CharValueTooShort element attribute or NumValueTooShort element attribute has been specified and the TrimLeadingBlanks element attribute has been specified with a value of Y. Leading white space is always removed to the first non-white space character.
- N** Does not remove all leading white space from the beginning of a simple element value before adding it to the output document.

Formatting notes for each document syntax type:

**XML** The default setting for this element attribute is N (do not remove leading blanks). All XML values have a minimum length 1. If the value is a string of blanks then it is reduced to a single blank.

#### **EDI Standard**

The default setting for this element attribute is N (do not remove leading blanks). The minimum length is specified in the definition of the Data Element.

#### **Data Format**

The default setting for this element attribute is N (do not remove leading blanks). The minimum length for a fixed length Data Format field is the same as the defined maximum length of the field. The minimum length of a character field in a comma delimited Data Format is zero.

## Commands

***TrimTrailingBlanks:*** Indicates how to treat trailing white space at the end of a simple element value. White space includes blanks, carriage returns, line feeds and tab characters. Valid values are:

- Y** Remove all trailing white space from the end of a simple element value to the minimum length of the simple element before adding it to the output document. The minimum length of the simple element is not considered when the CharValueTooShort element attribute or NumValueTooShort element attribute has been specified and the TrimTrailingBlanks element attribute has been specified with a value of “Y”. Trailing white space is always removed to the last non-white space character in this situation.
- N** Do not remove all trailing white space from the end of a simple element value before adding it to the output document.

Formatting notes for each document syntax type:

**XML** The default setting for this element attribute is Y (remove trailing blanks). All XML values have a minimum length 1. If the value is a string of blanks then it is reduced to a single blank.

### **EDI Standard**

The default setting for this element attribute is Y (remove trailing blanks). The minimum length is specified in the definition of the Data Element.

### **Data Format**

The default setting for this element attribute is “N” (do not remove trailing blanks). The minimum length for a fixed length Data Format field is the same as the defined maximum length for the field. The minimum length of a character field in a comma delimited Data Format is one.

***SuppressZeroValues:*** Indicates whether or not to output the value of a simple element declared to be of a numeric type, if it has a value of zero. Valid values are:

- Y** Suppresses zero values and does not write the value of the simple element to output document if it has a value of zero. Compound elements with this value for the SuppressZeroValues element attribute are not written to the output document if all of the elements within the compound element contain zero values.
- N** Does not suppress zero values writes the value of the simple element to the output document even when it has a value of zero.

Formatting notes for each document syntax type:

**XML** The default setting for this element attribute is N (do not suppress zero values). A value is considered to be numeric if the value contains all numbers or numbers with a decimal. The evaluation to determine if an XML value is numeric is performed after any optional operations to remove leading or trailing blanks. By default, trailing blanks are automatically removed for XML values while leading blanks are not automatically removed.

### **EDI Standard**

The default setting for this element attribute is Y (suppress zero values).



Mandatory Data Elements with a zero value are written to the output document regardless of the setting for this element attribute. A Data Element is considered to be numeric if its data type is real or numeric

### Data Format

The default setting for this element attribute is N (do not suppress zero values). A Data Format field is considered to be numeric if its data type is real, numeric, integer, binary, hexadecimal, packed decimal, or zoned decimal. This element attribute has the effect of moving no value for fixed position Data Format output files. This means that numeric fields have blank values in the output record. This element attribute affects the output of a field for comma separated values in a Data Format output file. If the suppressed zero value is the last Field in the Data Format Record then the terminating comma for the last Field is not written.

**SuppressEmptyElements:** Indicates whether or not to output the value of a simple element if it has no value. An element is considered to have no value when it has a length of zero or it contains only white space. White space includes blanks, carriage returns, line feeds and tab characters. Valid values are:

- Y** Suppresses empty values. Specifies that a simple element is not written to the output document if it has no value. Compound elements are not written to the output document if any element within the compound element does not contain a value.
- N** Does not suppress empty values. The element is written to the output document even if it has no value or all the elements contained within it have no value.

Formatting notes for each document syntax type:

**XML** The default setting for this element attribute is N (do not suppress empty elements).

### EDI Standard

The default setting for this element attribute is Y (suppress empty elements). Mandatory Composite Data Elements and mandatory Data Elements are written to the output document regardless of the setting for this element attribute. When the setting is Y for:

- A Segment (mandatory or optional) then the Segment is not written to the output document if all Data Elements within the Segment are empty.
- An optional Composite Data Element then the Composite Data Element is not written to the output document if all Data Elements within the Composite Data Element are empty. Only the required delimiters are written.
- An optional Data Element then the Data Element is not written to the output document. A delimiter is written if required by the EDI Standard. If the value is N on a Data Element, then the Data Element is written to the output document even when it contains no value. This means that the terminator for the Data Element preceding the Data Element with the N value for its SuppressEmptyElements element attribute is written.

### Data Format

The default setting for this element attribute is N (do not suppress empty elements). If the SuppressEmptyElements element attribute is specified on a Data Format Record with a value of Y then the Data Format record is not written to the output document if none of the fields within the record contain a value. If the SuppressEmptyElements element attribute is specified on a Data Format loop with a value of Y, then the Data Format records within the loop are not written to the output document unless one or more fields within the record contains a value. This assumes that no Data Format records within the loop have the SuppressEmptyElements element attribute for the record set to N. The SuppressEmptyElements element attribute has no effect for a fixed position Data Format field. It does affect the output of a field in a comma separated values Data Format output file.

**NumValueTooShort:** This element attribute indicates how to handle numeric values that are shorter than the minimum length defined for the simple element. It is only useful when the simple element is defined with a numeric data type. Valid values are:

- RB** Right justify the value and pad on the left with blanks to the minimum length of the simple element.
- RZ** Right justify the value and pad on the left with zeros to the minimum length of the simple element.
- LB** Left justify the value and pad on the right with blanks to the minimum length of the simple element.
- LZ** Left justify the value and pad on the right with zeros to the minimum length of the simple element.
- IG** Ignore minimum length specification for the simple element. No left or right justification is performed. The value of the simple element is written to the output document without insuring the value meets the minimum length specification for the simple element.

**Note:** The right and left justify operations do not remove leading or trailing zeros, blanks, or other white space. Use the various TrimLeadingBlanks, TrimTrailingBlanks, TrimLeadingZeros, and TrimTrailingZeros element attributes to ensure you get the justification required. The minimum length of the simple element is not considered when using this attribute and the various trimming attributes.

Formatting notes for each document syntax type:

**XML** All elements in XML have a default minimum length of one. Therefore this element attribute never has any affect on an XML element.

### EDI Standard

The default value is RZ for Data Elements defined as a numeric data type. All other Data Elements have a default value of IG. A Data Element is considered to be numeric if its data type is real or numeric.

### Data Format

The default setting for this element attribute is RZ for numeric Data Format

fields. LB is the default value for non-numeric Data Format Field. The minimum length for a fixed length Data Format field the same as the defined maximum length of the field. The minimum length of a field in a comma delimited Data Format is one unless dictated otherwise by the data type. A Data Format field is considered to be numeric if its data type is real, numeric, integer, binary, hexadecimal, packed decimal, or zoned decimal.

**CharValueTooShort:** This element attribute indicates how to handle character values that are shorter than the minimum length defined for the simple element. Elements that are defined as a numeric data type that is written as characters can utilize this element attribute effectively. Valid values are:

- RB** Right justify the value and pad on the left with blanks to the minimum length of the simple element.
- RZ** Right justify the value and pad on the left with zeros to the minimum length of the simple element.
- LB** Left justify the value and pad on the right with blanks to the minimum length of the simple element.
- LZ** Left justify the value and pad on the right with zeros to the minimum length of the simple element.
- IG** Ignore minimum length specification for the simple element. No left or right justification is performed. The value of the simple element is written to the output document without insuring the value meets the minimum length specification for the simple element.

**Note:** The right and left justify operations do not remove leading or trailing zeros, blanks, or other white space. Use the TrimLeadingBlanks, TrimTrailingBlanks, TrimLeadingZeros, and TrimTrailingZeros element attributes to ensure you get the justification required. The minimum length of the simple element is not considered when using this attribute and the various trimming attributes.

Formatting notes for each document syntax type:

**XML** All elements in XML have a default minimum length of one. Therefore this element attribute does not have any affect on an XML element.

### **EDI Standard**

The default value is RZ for Data Elements defined as a numeric data type. All other Data Elements have a default value of IG. A Data Element is considered to be numeric if its data type is real or numeric.

### **Data Format**

The default setting for this element attribute is RZ for numeric Data Format Fields. LB is the default value for non-numeric Data Format field. The minimum length for a fixed length Data Format field is the same as the defined maximum length of the field. The minimum length of a field in a comma delimited Data Format is one unless dictated otherwise by the data type. A Data Format field is considered to be numeric if its data type is real, numeric, integer, binary, hexadecimal, packed decimal, or zoned decimal.

### Processing order

The following lists the order in which the element attributes are applied to an element as it is written to an output document:

- TrimLeadingBlanks
- TrimTrailingBlanks

If the value is numeric:

- TrimTrailingZeros
- TrimLeadingZeros
- NumValueTooShort (LB)  
Adjust left blank pad
- NumValueTooShort (LZ)  
Adjust left zero pad
- NumValueTooShort (RB)  
Adjust right blank pad
- NumValueTooShort (RZ)  
Adjust right zero pad

If the value is not numeric:

- CharValueTooShort (LB)  
Adjust left blank pad
- CharValueTooShort (LZ)  
Adjust left zero pad
- CharValueTooShort (RB) Adjust right blank pad
- CharValueTooShort (RZ)  
Adjust right zero pad

Formatting notes for each document syntax type:

**XML** A value is considered to be numeric if the value contains all numbers or numbers with a decimal. The evaluation to determine if an XML value is numeric is performed after any optional operations to remove leading or trailing blanks. By default, trailing blanks are automatically removed for XML values while leading blanks are not automatically removed.

#### EDI Standard

A Data Element is considered to be numeric if its data type is real or numeric.

#### Data Format

A Data Format Field is considered to be numeric if its data type is real, numeric, integer, binary, hexadecimal, packed decimal, or zoned decimal.

### Example 1

The following command alters the default setting for the SuppressZeroValues element attribute for all simple elements within the target document. The command indicates that simple elements are written to the output document even when they contain a zero value unless otherwise specified.

```
SetElementAttribute (TargetRoot, "SuppressZeroValues", "N")
```

The following command alters the setting for the SuppressZeroValues element attribute for the M8 simple element within the target document. The command indicates that the M8 simple element is written to the output document if it contains a zero value.

```
SetElementAttribute (\ADTBXML\M8\\, "SuppressZeroValues", "Y")
```

Formatting notes for each document syntax type:

**XML** For the examples in Table 24, assume that the XML element M803 contains an attribute element named desc and an element containing the value associated with the M803 element. It is assumed that the SuppressEmptyElements element attribute is set to N for the M803 element.

Table 24. Format examples for SuppressZeroValues

| Example |                    | Attribute element | Value element | Output result     |
|---------|--------------------|-------------------|---------------|-------------------|
| 1       | Element value      | 0                 | 0             | <M803 desc="0"/>  |
|         | SuppressZeroValues | N                 | N             |                   |
| 2       | Element value      | 0                 | 0             | <M803 desc="0"/>  |
|         | SuppressZeroValues | N                 | Y             |                   |
| 3       | Element value      | 0                 | 0             | <M803>0</M803>    |
|         | SuppressZeroValues | Y                 | N             |                   |
| 4       | Element value      | 0                 | 0             | <M803/>           |
|         | SuppressZeroValues | Y                 | Y             |                   |
| 6       | Element value      | 12                | 0             | <M803 desc="12"/> |
|         | SuppressZeroValues | Y                 | Y             |                   |
| 7       | Element value      | 0                 | 82            | <M803>82</M803>   |
|         | SuppressZeroValues | Y                 | Y             |                   |

### Data Format

If a Record contains four fields, RECORDID, TYPE, QUANTITY, and DESCRIPTION with the values "RECID", "ABC", 0, and "My Item Order" respectively, then the comma separated values Record appears as follows when the SuppressZeroValues element attribute is set to "N" for the third Field: "RECID","ABC",0,"My Item Order" The Record appears as follows when the SuppressZeroValues element attribute is set to "Y" for the third Field: "RECID","ABC",,"My Item Order" If the suppressed zero value is the last Field in the Data Format Record then the terminating comma for the last Field is not written

### Example 2

The following command alters the setting for the SuppressEmptyElements element attribute for the LOOPX02 compound element within the target document. The command indicates that the LOOPX02 compound element is not written to the output document if all simple elements within it contain no significant value.

```
SetElementAttribute (\DATHDR_LOOP2\LOOPX02\\, "SuppressEmptyElements", "Y")
```

## Commands

**XML** For the following formatting examples, assume XML element M803 contains an attribute element named desc and an element containing the value associated with the M803 element. It is assumed that the SuppressEmptyElements element attribute is set to N for the M803 element.

Table 25. Format examples for SuppressZeroValues

| Example |                      | Attribute element | Value element | M803 | Output result                    |
|---------|----------------------|-------------------|---------------|------|----------------------------------|
| 1       | Element value        | " "               | " "           | —    | <M803 desc=" " ><br></M803>      |
|         | SuppressEmptyElement | N                 | N             | N    |                                  |
| 2       | Element value        | " "               | " "           | —    | <M803 desc=" " />                |
|         | SuppressEmptyElement | N                 | Y             | N    |                                  |
| 3       | Element value        | " "               | " "           | —    | <M803> </M803>                   |
|         | SuppressEmptyElement | Y                 | N             | N    |                                  |
| 4       | Element value        | " "               | " "           | —    | <M803/>                          |
|         | SuppressEmptyElement | Y                 | Y             | N    |                                  |
| 6       | Element value        | "ABC"             | " "           | —    | <M803 desc="ABC"/>               |
|         | SuppressEmptyElement | Y                 | Y             | N    |                                  |
| 7       | Element value        | " "               | "DEF"         | —    | <M803>DEF</M803>                 |
|         | SuppressEmptyElement | Y                 | Y             | N    |                                  |
| 7       | Element value        | "ABC"             | " "           | —    | <M803 desc="ABC"/>               |
|         | SuppressEmptyElement | Y                 | Y             | Y    |                                  |
| 8       | Element value        | " "               | " "           | —    | M803 XML element is not written. |
|         | SuppressEmptyElement | Y                 | Y             | Y    |                                  |

### Data Format

If a Record contains four fields, RECORDID, TYPE, DESCRIPTION, and QUANTITY with the values "RECID", "ABC", "" and 0 respectively, then the comma separated values record appear as follows when the SuppressEmptyElements element attribute is set to N for the third Field:

```
"RECID", "ABC", "", 0
```

The record will appear as follows when the SuppressZeroValues element attribute is set to Y for the third Field:

```
"RECID", "ABC", , 0
```

If the suppressed empty value is the last field in the Data Format Record then the terminating comma for the last field will not be written.

### SetNamespace

If the **SetNamespace** command is included, an attribute of the following form will be created on the root element of the XML output:

```
xmlns:prefix="URI"
```

The specified URI is looked up in the namespace table, and the prefix is taken from that entry. For example: `xmlns("http://www.ibm.com/schema/example")` would generate the following (assuming the namespace table defines `xmp` as the prefix for namespace `http://www.ibm.com/schema/example`):

```
xmlns:xmp="http://www.ibm.com/schema/example"
```

If there is more than one occurrence of this command, then an attribute will be created for each command.

`SetNamespace` has the following format:

```
SetNamespace(URI)
```

Where:

**URI** is a URI from the namespace table.

### SetNoNSSchemaLocation

If this command is included, an attribute of the following form will be created on the root element of the XML output:

```
xsi:noNamespaceSchemaLocation="location"
```

For example:

```
noNamespaceSchemaLocation("myschema.xsd")
```

Would generate the following:

```
xsi:noNamespaceSchemaLocation="myschema.xsd"
```

**Note:** The `xsi` prefix will be changed if you have a different prefix specified in the namespace table for `http://www.w3.org/2001/XMLSchema-instance`. If you do not have a namespace table entry for `http://www.w3.org/2001/XMLSchema-instance` the default prefix `xsi` will be used. An appropriate `xmlns:prefix=http://www.w3.org/2001/XMLSchema-instance` attribute will automatically be generated when this command is used.

If there is more than one occurrence of this command, only the last value is used.

`SetNoNSSchemaLocation` has the following format:

```
SetNoNSSchemaLocation(location)
```

Where:

**location** is a character expression that indicates the schema location for the output.

### SetSchemaLocation

If this command is included, an attribute of the following form will be created on the root element of the XML output:

```
xsi:schemaLocation="URI location"
```

## Commands

The specified URI is looked up in the namespace table, and the location is taken from that entry. For example:

```
SetSchemaLocation("http://www.ibm.com/schema/example")
```

Would generate the following (assuming the namespace table defines example.xsd as the location for namespace http://www.ibm.com/schema/example):

```
xsi:schemaLocation="http://www.ibm.com/schema/example example.xsd"
```

**Note:** The xsi prefix will be changed if you have a different prefix specified in the namespace table for http://www.w3.org/2001/XMLSchema-instance. If you do not have a namespace table entry for http://www.w3.org/2001/XMLSchema-instance the default prefix xsi will be used. An appropriate xmlns:prefix=http://www.w3.org/2001/XMLSchema-instance attribute will automatically be generated when this command is specified.

This command might be specified multiple times in the map. If there is more than one occurrence of this command, then a namespace/location pair will be created for each command.

SetSchemaLocation has the following format:

```
SetSchemaLocation(URI)
```

Where:

**URI** is a URI from the namespace table.

## SetProperty

The **SetProperty** command is used to set a special processing property of the target message. Various special properties are defined to control things such as the EDI envelope fields or the XML prolog. The **SetProperty** command uses the following format:

```
SetProperty(char propertyName, char propertyValue)
```

Where:

- |                      |                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>propertyName</b>  | The name of the document property you want to set. See Document properties for additional information about document properties and a complete list of properties that you can set. This is typically a string constant, but can be a variable, a path identifying a simple element in the source document definition, or an expression that evaluates to a character string. |
| <b>propertyValue</b> | The value the document property is set to. This can be a numeric or string constant, variable, path identifying a simple element in the source document definition, or an expression.                                                                                                                                                                                         |



See “Message properties” on page 193 for a list of the property names that can be specified.

---

## Functions

All functions take zero or more arguments as input and return a value. The data type of the return value, as well as the number and data types of the arguments varies from one function to the next. Appropriate type conversions are done implicitly if needed (and possible), just as they are done for the expressions.

Some functions have optional parameters. If the optional parameters are omitted from the function call, a default value is used for that argument.

Most functions can take an expression as an argument, as long as the result of the expression is (or can be converted to) the correct data type. For example, the Char function converts a value to a character string. The command:

```
Var1 = Char (1 + 2)
```

Is equivalent to:

```
Var1 = Char (3)
```

The only time you cannot use an expression as an argument is when the argument is identified as a source or target path. For example, you cannot pass an expression to the Created function.

The input arguments for a function are never modified by the function. The return value and argument list for each of the functions is described in the following sections.

The function names are not case sensitive. For example, the function name Char is the same as CHAR.

## Char

The Char function returns a character representation of a numeric value. The Char function uses the following format:

```
char Char(real value)
```

Where:

**value** Is the value to be converted to a character value. It can be of data type real or int. (Data type character can be used, but does not cause any conversion.)

## Results

The character representation of *value*.

This function is not normally needed, because the conversion is generally done implicitly. However, it can be used to force the conversion to a character value, or to clarify when the conversion is done.

## Commands

### Concat

The Concat function concatenates one character string to another. The Concat function uses the following format:

```
char Concat(char value1, char value2)
```

Where:

**value1** Is the value that is appended to.

**value2** Is the value that is appended to the end of *value1*.

### Results

The concatenated string.

### Example

```
chVar = Concat("abc", "def")
```

Would set chVar to "abcdef".

### Created

The Created function is used to determine if the specified path was created in the target data. The Created function uses the following format:

```
boolean Created(targetPath)
```

Where:

**targetPath** Identifies the path in the target document that is being checked.

### Results

**True** When the specified path has been created in the target data

**False** When it has not been created in the target data.

This function can only be used with paths in the target document definition. If the element is within a repeating compound element (such as a repeating loop or segment), then only the current (or most recent) iteration is searched for the *targetPath*.

This function can only be used with paths that can occur in the target document definition.

### Example

```
If (!Created(\T 2\L 2300\S CLM 130\C C023 5\))
 mapping command
EndIf
```

In this example, if path “\T 2\L 2300\S CLM 130\C C023 5\” has not been created, *mapping command* is executed by the translator.

## Date

The Date function returns the system date as a character string in the format `yyyymmdd`. The Date function uses the following format:

```
char Date()
```

### Results

The current system date in the format `yyyymmdd`.

## DateCnv

The DateCnv function is used to assist in converting one date format to another. The DateCnv function uses the following format:

```
char DateCnv(char sourceDate, char frommask, char tomask)
```

The actual syntax requires quotation marks. For example:

```
Mapping Path = DateCnv('20021216','CCYYMMDD','CCYY-MM-DD')
```

Where:

- sourceDate** Is the string that contains the source date.
- frommask** Is the mask that identifies the format of the date in *sourceDate*.
- tomask** Is the mask that identifies the format of the date required in the result string.

### Results

The converted date as a character string.

The mask format for the DateCnv function is the same as the mask format used in send and Receive maps. “Date conversion special operators” on page 263 for information about the values that can appear in the from and to mask.

## Exit

The any-to-any mapping function is called **Exit**. **Exit** can be used within transformation maps to invoke a field exit and will return a string value. Results that are returned can be used to update a variable or target a simple element in an assignment statement. The **Exit** function uses the following format:

```
char Exit(char exitname[, char parameter[, char parameter[, ...]]])
```

Where:

- result** A string returned by the user exit
- exitname** A string expression that results in the **User Exits** profile member name that contains the function information to be executed.
- parameter** Evaluates to a string that will be passed to the field exit. Parameter cannot evaluate to a simple element in the target data. Up to 4 parameters can be passed to the **Exit** function.

## Commands

You must define field exits in the **User Exits** profile. If the exit is not properly defined, an error is issued. Zero to four parameters can be passed to the exit and it is the job of the field exit to determine how many parameters to expect.

## Find

The Find function is used to determine if a string is contained within another string. The Find function uses the following format:

```
real Find(char toSearch, char searchVal, real startPos)
```

Where:

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <b>toSearch</b>  | Is the string to be searched.                                                     |
| <b>searchVal</b> | Is the string that is to be searched for within <i>toSearch</i> .                 |
| <b>startPos</b>  | Is the starting position within <i>toSearch</i> where the search will be started. |

## Results

The position where *string* occurs in the simple element or variable.

Zero is returned when:

- *searchVal* is not found within *toSearch*
- The *toSearch* value has less than *startPos* characters
- *toSearch* is empty

*startPos* is one-based, so a *startPos* of one will start the search from the beginning of the string, a position of two begins the search at the second character, and so on.

## Examples

Assuming *var1* contains the string "ABCDEFGH",

```
Find(var1, "CDE", 1) will return 3
```

```
Find(var1, "CDE", 3) will return 3
```

```
Find(var1, "CDE", 4) will return 0
```

```
Find(var1, "CDE", 0) will cause an error to be issued
```

## Found

Found is used to determine if a specified path exists in the source data. The Found function uses the following format:

```
boolean Found(sourcePath)
```

Where:

|                   |                                            |
|-------------------|--------------------------------------------|
| <b>sourcePath</b> | Identifies the path that is being checked. |
|-------------------|--------------------------------------------|

## Results

**True** When the specified path is found in the source data

**False** When it is not in the source data.

This function can only be used with paths in the source document definition. If the element is within a repeating compound element (such as a repeating loop or segment), then only the current iteration is searched for the *sourcePath*.

## Example

```
If (!Found(\T 2\L 2300\S CLM 130\C C023 5\\))
 mapping command
EndIf
```

In this example, if path “\T 2\L 2300\S CLM 130\C C023 5\\” is not found, *mapping command* is executed by the translator.

## GetProperty

The GetProperty function is used to get a property of the source message. This can be used to retrieve information such as EDI envelope header elements. The GetProperty function uses the following format:

```
char GetProperty(char propertyName)
```

Where:

**propertyName** Is the name of the property you want to retrieve. See “Message properties” on page 193 for a list of properties that you can retrieve.

## Results

The value associated with the specified message property.

If the propertyName is not set in the source message, an empty string is returned.

## Example

```
var1 = GetProperty("ISA04")
```

Will set var1 to the value that was in the ISA04 element in the X12 ISA segment.

## HexEncode

The HexEncode function is used to encode binary data to a character format. The HexEncode function uses the following format:

```
char HexEncode(binary binValue)
```

Where:

**binValue** Is the binary value to be encoded.

## Commands

### Results

An encoded character string that represents the binary data.

Each byte of the binary value will be encoded as two characters in the resulting string. For example, if the input is a 4 byte binary value: 0x01020A0B, the result would be the 8-character string: "01020A0B".

## HexDecode

The HexDecode function is used to decode a character string that contains encoded binary data. The HexDecode function uses the following format:

```
binary HexDecode(char encodedStr)
```

Where:

**encodedStr** Is an encoded string.

### Results

A binary value derived from the encoded character string.

Each byte of the binary value will be derived from two characters in the encoded string. For example, if the input is the 8-character string: "01020A0B", the result would be a 4 byte binary value: 0x01020A0B.

## IsEmpty

The IsEmpty function is used to determine if a character string contains any data. The IsEmpty function uses the following format:

```
boolean IsEmpty(char value)
```

Where:

**value** Is the value to check.

### Results

**True** When *value* is empty (set to an empty string: "").

**False** When *value* contains data.

If the *value* is a source element, this will also return **True** if the element is not found. The expression:

```
IsEmpty(value)
```

is equivalent to the following string comparison expression:

```
(StrComp(value, "") = 0)
```

### Example

```
If (not IsEmpty(var1))
 mapping command
EndIf
```

In this example, if *var1* contains a value other than an empty string, *mapping command* is executed by the translator.

## Left

The Left function is used to obtain the first few characters from a string. The Left function uses the following format:

```
char Left(char stringIn, real length)
```

Where:

**stringIn** Is the input string that characters are copied from.

**length** Is the number of characters to be copied.

### Results

A character string containing the leftmost characters of *stringIn*.

Only available characters from *stringIn* will be copied. Therefore, the returned string will contain *length* characters unless *stringIn* has less than *length* characters. See “Example 2.”

#### Example 1

```
var1 = left("ABCDEFGH",3)
```

After this code has been executed, *var1* will equal “ABC”.

#### Example 2

```
var1 = left("ABCD",6)
```

After this code has been executed, *var1* will equal “ABCD” because there are not enough characters in the *stringIn* argument to fulfill the request.

## Length

The Length function is used to determine the length of a string. The Length function uses the following format:

```
real Length(char value)
```

Where:

**value** Is a string, usually a variable or a simple element in the source document definition.

### Results

The length of the character string value.

## Commands

### Example

```
If (Length(var1) < 5)
 mapping command
EndIf
```

In this example, if *var1* has a length that is less than 5, *mapping command* is executed by the translator.

## Lower

The Lower function is used to convert a string to lowercase. The Lower function uses the following format:

```
char Lower(char value)
```

Where:

**value** Is a string, usually a variable or a simple element in the source document definition.

### Results

A copy of *value*, with all characters converted to lowercase.

### Example

```
var1 = Lower("ABC")
```

In this example, *var1* will be set to "abc".

## Number

The Number function converts a character value to a real value. The Number function uses the following format:

```
real Number(char value)
```

Where:

**value** Is the character value to be converted to a real value. (Data type real or int can be used, but do not cause any conversion.)

### Results

The (data type) real representation of *value*.

This function is not normally needed, because the conversion is generally done implicitly. However, it can be used to force the conversion to a numeric value, or to clarify when the conversion is done.

## NumFormat

The NumFormat function formats a real number or integer as a character string using a specified number of decimal positions. Unused digits are either truncated or rounded.

The NumFormat function uses the following format:

```
char NumFormat(real value, real decimals[, char flag])
```



Where:

|                 |                                                                                                                                                                                                                                        |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>value</b>    | Is the number that will be formatted as a character string.                                                                                                                                                                            |
| <b>decimals</b> | Is the number of decimal positions that are included in the returned value.                                                                                                                                                            |
| <b>flag</b>     | Indicates whether unused digits are rounded or truncated. Specify “ROUND” to round unused digits or “TRUNCATE” to truncate unused digits. These can be shortened to “R” or “T”. If no flag is specified, the value default is “ROUND”. |

## Results

The character representation of *value*, containing the specified number of decimal places.

### Example 1

```
var1 = NumFormat(1234.567, 2, "T")
```

This example will set *var1* to “1234.56”.

### Example 2

```
var1 = NumFormat(1234.567, 2, "R")
```

This example will set *var1* to “1234.57”.

## Occurrence

Use the Occurrence function to obtain the current occurrence number of:

- The current repeating compound element or simple element
- A specific repeating compound element or simple element

Occurrence can be shortened to Occur. The Occurrence function has the following format:

```
real Occurrence([sourcepath])
```

Where:

**sourcepath** Is the path that refers to a compound or simple element in the source document definition. If no *sourcepath* is specified, the current compound or simple element in the source document is assumed.

If this function is used in a target-based map, you must include the source element as a parameter. The source element must be the current source domain, or an ancestor (for example, parent or grandparent) of the current source domain. If you omit the *sourcepath*, the compiler command processor issues an error.

For source-based maps, the source element parameter continues to be optional (if omitted, the current source element is assumed).

## Commands

### Results

The current occurrence number of the specified element.

When Occurrence is specified with no parameters, it will return the occurrence number of the current compound or simple element. When *sourcePath* is specified, it will return the current occurrence number of the specified repeating compound element or simple element. The specified element must be the current element, a parent of the current element, or one of the other elements on the path to the current element (such as the grandparent of the current element).

### Example 1

```
If (Occurrence (\Table 2\10 M PO1 Loop\)) = 2)
 mapping command
EndIf
```

In this example, if this is the second occurrence of the PO1 loop, then *mapping command* will be executed by the translator.

### Example 2

```
If (Occurrence() != 1)
 mapping command
EndIf
```

In this example, if this is not the first occurrence of the current source element, *mapping command* is executed by the translator.

## Overlay

The Overlay function is used to overlay a portion of a string with data from another string. The Overlay function uses the following format:

```
char Overlay(char string1, char string2, real position [,
real length])
```

Where:

- |                 |                                                                                                                                                                                                                                                          |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>string1</b>  | Evaluates to a string that will be overlaid with data from string2.                                                                                                                                                                                      |
| <b>string2</b>  | Evaluates to a string that will overlay data in string1.                                                                                                                                                                                                 |
| <b>position</b> | Indicates the position where the overlay will begin in <i>string1</i> .                                                                                                                                                                                  |
| <b>length</b>   | Is an optional parameter that indicates the number of characters to be overlaid in <i>string1</i> . If omitted or less than 0, all of the characters following <i>position</i> will be overlaid until all characters from <i>string2</i> have been used. |

### Results

The resulting string *position* is one-based, so a position of one will begin overlaying characters at the beginning of *string1*, a position of two begins overlaying at the second

character, and so on. If the length of *string1* has less than *position* characters, then blanks will be used to fill the positions between the end of *string1* and the starting position.

### Example

```
var1 = overlay("ABCDEFGF", "WXYZ", 3, 3)
```

After this code has been executed, *var1* will equal "ABWXYFG".

## Right

The Right function is used to obtain the last few characters from a string. The Right function uses the following format:

```
char Right(char stringIn, real length)
```

Where:

**stringIn**            Is the input string that characters are copied from.

**length**             Indicates the number of characters to be copied.

### Results

A character string containing the rightmost characters of *stringIn*.

Only available characters from *stringIn* will be copied. Therefore, the returned string will contain *length* characters unless *stringIn* has less than *length* characters. See "Example 2."

### Example 1

```
var1 = Right("ABCDEFGF",3)
```

After this code has been executed, *var1* will equal "EFG".

### Example 2

```
var1 = Right("ABCD",6)
```

After this code has been executed, *var1* will equal "ABCD" because there are not enough characters in the *stringIn* argument to fulfill the request.

## Round

The Round function rounds the input value to the specified number of decimal places. The Round function uses the following format:

```
real Round(real value, real decimals)
```

Where:

**value**                Is the value that will be rounded.

**decimals**            Is the number of decimals to round to.

## Commands

### Results

A real number rounded to the specified number of decimal places.

For instance, `Round(4321.556, 2)` will return a result of 4321.560000. `Round(4321.556, 0)` will return a result of 4322.000000.

**Note:** This function returns a numeric (real) value. If the return value will be assigned to a character string and trailing zeros are to be removed, the `NumFormat` function is used.

## StrComp

The `StrComp` function is used to compare two character strings. The `StrComp` function uses the following format:

```
real StrComp(char string1, char string2)
```

Where:

**string1**            Is the first string to be compared.

**string2**            Is the second string to be compared.

### Results

-1        If *string1* < *string2*  
0        If the strings are equal  
1        If *string1* > *string2*

### Example

```
If (StrComp(var1, "ABC") = 0)
 mapping command
EndIf
```

In this example, if *var1* is "ABC", *mapping command* is executed by the translator.

## StrCompI

The `StrCompI` function is used to compare two character strings without regard to case. The `StrCompI` function uses the following format:

```
real StrCompI(char string1, char string2)
```

Where:

**string1**            Is the first string to be compared.

**string2**            Is the second string to be compared.

### Results

-1        If *string1* < *string2* (case insensitive)  
0        If the strings are equal (case insensitive)  
1        If *string1* > *string2* (case insensitive)

**Example**

```
If (StrCompI(var1, "ABC") = 0)
 mapping command
EndIf
```

In this example, if *var1* is "ABC", "abc", "Abc", *mapping command* is executed by the translator.

**StrCompN**

The StrCompN function is used to compare the first *length* characters of two character strings. The StrCompN function uses the following format:

```
real StrCompN(char string1, char string2, real length)
```

Where:

**string1**            Is the first string to be compared.  
**string2**            Is the second string to be compared.  
**length**             Is the number of characters to compare.

**Results**

**-1**            If the first *length* characters of *string1* < the first *length* characters of *string2*  
**0**             If the first *length* characters of the two strings are equal  
**1**             If the first *length* characters of *string1* > the first *length* characters of *string2*

**Example**

```
If (StrCompN(var1, "ABC", 3) = 0)
 mapping command
EndIf
```

In this example, if the first 3 characters of *var1* are "ABC", then *mapping command* will be executed by the translator.

**StrCompNI**

The StrCompNI function is used to compare the first *length* characters of two character strings without regard to case. The StrCompNI function uses the following format:

```
real StrCompNI(char string1, char string2, real length)
```

Where:

**string1**            Is the first string to be compared.  
**string2**            Is the second string to be compared.  
**length**             Is the number of characters to compare.

**Results**

**-1**            If the first *length* characters of *string1* < the first *length* characters of *string2*  
(case insensitive).

## Commands

- 0 If the first *length* characters of the two strings are equal (case insensitive).
- 1 If the first *length* characters of *string1* > the first *length* characters of *string2* (case insensitive).

### Example

```
If (StrCompNI(var1, "ABC", 3) = 0)
 mapping command
EndIf
```

In this example, if the first 3 characters of *var1* are "ABC", "abc", "Abc", and so on, *mapping command* is executed by the translator.

## SubString

The SubString function is used to obtain a portion of a string. The SubString function uses the following format:

```
char SubString(char string, real position [, real length])
```

Where:

- string** Is the string value.
- position** Indicates the starting position of the substring within the *string*.
- length** Is an optional parameter that indicates the number of characters to be copied into the substring. If this argument is omitted or the length is less than 0, all remaining characters to the end of the string are copied.

### Results

The resulting character string returned by the function.

*position* is one-based, so a *position* of one will start copying characters from the beginning of *string*, a position of two begins copying data from the second character, and so on

Only available characters from the *string* value will be copied. Therefore, the substring will contain *length* characters unless *string* has less than *position* plus *length* minus one characters. See "Example 2." If the *string* value has less than *position* characters, an empty string will be returned.

### Example 1

```
var1 = substring("ABCDEFGH",3,3)
```

After this code has been executed, *var1* will equal "CDE".

### Example 2

```
var1 = substring("ABCD",3,3)
```

After this code has been executed, *var1* will equal “CD” because there are not enough characters in the *string* argument to fulfill the request.

## Time

The Time function returns the system time as a character string in the format hhmmss. The Time function uses the following format:

```
char Time()
```

### Results

The system time as a character string in the format hhmmss.

## Translate

The Translate function attempts to locate a specified string in a translation table. If the string is found in the table, the corresponding value in the table is returned. The Translate function uses the following format:

```
char Translate(char table, char direction, char valueIn [,
boolean error[, char default]])
```

Where:

|                  |                                                                                                                                                                                                                                                       |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>table</b>     | Is the string that identifies the translation table.                                                                                                                                                                                                  |
| <b>direction</b> | Is either “SOURCE” or “TARGET”. This can be shortened to either “S” or “T”.                                                                                                                                                                           |
| <b>valueIn</b>   | Is the string value that will be looked up in the translation table.                                                                                                                                                                                  |
| <b>error</b>     | Indicates whether a not found condition will result in a warning message. If <b>True</b> , a warning message will be issued if the value is not found in the table. If <b>False</b> , no warning message will be issued. The default is <b>True</b> . |
| <b>default</b>   | Is a string that provides a default value that is returned when <i>valueIn</i> is not found in the translation table.                                                                                                                                 |

### Results

The resulting character string.

*table* must be a valid *Forward Translation* table. If *table* does not exist, a warning will be issued and it will be treated as a not found condition.

A translation table contains two values for each entry in the table, one called the *Local Value* and the other called the *Standards or Trading Partner Value*. When *direction* is specified as “SOURCE”, an attempt is made to find *valueIn* in the *Local Value* column in the table. If it is found, the value from the corresponding *Standards or Trading Partner Value* column is returned by the function. When *direction* is specified as “TARGET”, an attempt is made to find *valueIn* in the *Standards or Trading Partner Value* column in the table. If it is found, the value from the corresponding *Local Value* column is returned by the function.

## Commands

In a *Forward Translation* translation table, the *Local Value* column must be unique. The *Standards or Trading Partner Value* column does not need be unique. A *Forward Translation* translation table used in the Translate command will always produce predictable results when SOURCE is specified as the direction. The results are also always predictable when TARGET is specified as the direction *if* the values in the *Standards or Trading Partner Value* column are unique. If the values are not unique in the *Standards or Trading Partner Value*, the results might not be predictable when TARGET is specified as the direction.

If *valueIn* is not found in the translation table, a warning message will be issued if *error* is **True**. Also, if *valueIn* is not found in the translation table, the *default* value will be returned if specified. If a *default* value is not specified, an empty string ("") is returned.

### Example

```
var1 = Translate("MYTABLE", "S", stringVar, False, "ABC")
```

This command will get the data contained in variable *stringVar* and attempt to locate that value in the *Local Value* column of translation table "MYTABLE". If the value is not found in the translation table, a warning message is not issued and *var1* will be set to "ABC". If the value is found in the translation table, *var1* will be set to the value in the corresponding *Standards or Trading Partner Value* column.

## TrimLeft

The TrimLeft function is used to remove unwanted leading characters from a string. The TrimLeft function uses the following format:

```
char TrimLeft(char value, [char trimList])
```

Where:

**value** Is the string value that is to be trimmed.

**trimList** Is an optional string value that includes all the characters that are to be removed. If this is not specified, all leading whitespace characters are removed.

### Results

The resulting character string.

All characters contained in *trimList* will be removed from the beginning of the *value* string. The operation ends when the first character not contained in *trimList* is encountered. If *trimList* is omitted, whitespace is removed from the beginning of the *value* string. Whitespace includes blanks, carriage returns, line feeds and tab characters.

### Example 1

```
var1 = TrimLeft(" ABCD")
```

In this example, *var1* will equal "ABCD" after the function has been executed.



## Example 2

```
var1 = TrimLeft("ZZYDABCD", "DYZ")
```

In this example, *var1* will equal “ABCD” after the function has been executed. All characters “D”, “Y” and “Z” will be removed from the beginning of *value* until an unspecified character is encountered.

## TrimRight

The TrimRight function is used to remove unwanted trailing characters from a string. The TrimRight function uses the following format:

```
char TrimRight(char value, [char trimList])
```

Where:

**value** Is a string value that is to be trimmed.

**trimList** Is an optional string value that includes all the characters that are to be removed. If this is not specified, all trailing whitespace characters are removed.

## Results

The resulting character string.

All characters contained in *trimList* will be removed from the end of the *value* string. The operation ends when the first character not contained in *trimList* is encountered. If *trimList* is omitted, whitespace is removed from the end of the *value* string. Whitespace includes blanks, carriage returns, line feeds and tab characters.

## Example 1

```
var1 = TrimRight("ABCD ")
```

In this example, *var1* will equal “ABCD” after the function has been executed.

## Example 2

```
var1 = TrimRight("ABCDZZY", "AYZ")
```

In this example, *var1* will equal “ABCD” after the function has been executed. All characters “A”, “Y” and “Z” will be removed from the end of the string until an unspecified character is encountered.

## Truncate

The Truncate function is used to truncate a number to the specified number of decimal places. The Truncate function uses the following format:

```
real Truncate(real value, real decimals)
```

Where:

**value** Is the value that will be truncated.

## Commands

**decimals** Is the number of decimals to truncate to.

### Results

A real number truncated to the specified number of decimal places.

The real number *value* is truncated to *decimals* decimal places - no rounding occurs. For instance, **Truncate**(4321.556, 2) will return a result of 4321.550000. **Truncate**(4321.556, 0) will return a result of 4321.000000.

**Note:** This function returns a numeric (real) value. If the return value will be assigned to a character string and trailing zeros are to be removed, the NumFormat function is used.

## Upper

The Upper function is used to change a string to upper case. The Upper function uses the following format:

```
char Upper(char value)
```

Where:

**value** Is a string, usually a variable or a simple element in the source document definition.

### Results

A copy of *value*, with all characters converted to uppercase.

### Example

```
var1 = Upper("abc")
```

In this example, *var1* will be set to "ABC".

## Validate

The Validate function attempts to locate a specified string in a validation table. The Validate function uses the following format:

```
boolean Validate(char table, char value [, boolean error])
```

Where:

**table** Is a string that identifies the validation table.

**value** Is a string value that will be looked up in the validation table.

**error** Indicates whether a not found condition will result in a warning message. If **True**, a warning message will be issued if the value is not found in the table. If **False**, no warning message will be issued. The default is **True**.

### Results

**True** If the string resulting from *value* is found in the validation table.

**False** Is returned if the string is not found in the validation table.

*table* must be a valid *Code List* table. If *table* does not exist, a warning message will be issued and it will be treated as a not found condition.

The *value* string is used to search the validation table. If the string is found in the validation table, **True** is returned by the function. If the string is not found in the validation table, **False** is returned.

### Example

```
If (Validate("MYTABLE",\Table 1\20 M BEG\1 M 353\))
 mapping command
Else
 Error(1, 5001, , "Invalid value specified")
EndIf
```

This code will get the data contained in simple element “\Table 1\20 M BEG\1 M 353\” and attempt to locate that value in the validation table “MYTABLE”. If the value is found in the validation table, the *mapping command* will be executed. If the value is not found in the validation table, an error is issued with error code 5001.

## Message properties

This section lists the message properties that you can access using the SetProperty command and GetProperty function:

- “Target document properties” on page 194
- “EDI envelope standard generic properties” on page 195
- “EDI envelope standard specific properties” on page 196

## Source document properties

The following properties are specific to source documents. They can be checked by using the GetProperty command.

|                    |                                                                                                                                                                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Alphanum</b>    | Inbound alphanumeric data validation table from the rule.                                                                                                                                                                                            |
| <b>Charset</b>     | Inbound character data validation table from the rule.                                                                                                                                                                                               |
| <b>GrpLvIFA</b>    | Group level functional acknowledgement only flag from the rule.                                                                                                                                                                                      |
| <b>InputMsgCnt</b> | Identifies the number of input messages processed.                                                                                                                                                                                                   |
| <b>LastMsg</b>     | A value of 'Y' indicates the current message being processed is the last input message is in the message flow.                                                                                                                                       |
| <b>MsgSplitCnt</b> | Identifies the number of XML documents split within each header/message/trailer split. The MsgSplitCnt property is set to zero until the last split message is processed. MsgSplitCnt property is reset with each new trailer/header identification. |
| <b>Process</b>     | The process value from the trading partner profile.                                                                                                                                                                                                  |
| <b>Thandle</b>     | Specifies the ID assigned by the system to a transaction when it is                                                                                                                                                                                  |

## Message properties

placed in the Document Store. To ensure uniqueness, the ID is a concatenation of the date, time, and a sequence number in format: YYYYMMDDHHMMSSnnnnnn.

|                    |                                               |
|--------------------|-----------------------------------------------|
| <b>ValErrLevel</b> | Inbound acceptable error level from the rule. |
| <b>ValLevel</b>    | Inbound validation level from the rule.       |
| <b>ValMap</b>      | Inbound validation map name from the rule.    |

## Target document properties

The following properties are specific to target documents. They can be set using the SetProperty command.

Setting these properties does not effect the current map, but might affect processing of the target document in later steps

|                     |                                                                                                                                                                                                     |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ACField</b>      | Substitutes the application control value established by the mappings of the AC field from the data format or the concatenation of the fields specified during creation of the transaction mapping. |
| <b>AckReq</b>       | Overrides the acknowledgement expected setting on the rule.                                                                                                                                         |
| <b>Alphanum</b>     | Overrides the outbound alphanumeric data validation table from the rule.                                                                                                                            |
| <b>Charset</b>      | Overrides the outbound character data validation table from the rule.                                                                                                                               |
| <b>CtlNumFlag</b>   | Overrides the control numbers by transaction id flag from the rule.                                                                                                                                 |
| <b>DIDocId</b>      | Document id value. This appears in the PRTFILE message when the output is written to a file or queue.                                                                                               |
| <b>DIProlog</b>     | Used to override the default XML prolog in the target document.                                                                                                                                     |
| <b>EdiDecNot</b>    | Overrides the EDI decimal notation character from the trading partner profile.                                                                                                                      |
| <b>EdiDeDIm</b>     | Overrides the EDI data element delimiter from the trading partner profile.                                                                                                                          |
| <b>EdiDeSep</b>     | Overrides the EDI repetition separator from the trading partner profile.                                                                                                                            |
| <b>EdiRIsChar</b>   | Overrides the EDI release character from the trading partner profile                                                                                                                                |
| <b>EdiSeDIm</b>     | Overrides the EDI subelement delimiter from the trading partner profile                                                                                                                             |
| <b>EdiSegDIm</b>    | Overrides the EDI segment delimiter from the trading partner profile                                                                                                                                |
| <b>EdiSegSep</b>    | Overrides the EDI segment id separator from the trading partner profile                                                                                                                             |
| <b>EncodeTarget</b> | Specifies an alternate encoding for target XML documents. Examples of other encodings that can be used are "UTF-16", "UTF-16LE", "UTF-16BE", "UCS-2", and "UTF-8".                                  |
| <b>EnvProfName</b>  | Overrides the envelope profile name from the rule.                                                                                                                                                  |

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| <b>EnvType</b>     | Overrides the envelope type from the rule.                               |
| <b>NetProfilId</b> | Overrides the network profile id from the trading partner profile        |
| <b>SegOutput</b>   | Overrides the segmented output flag from the trading partner profile     |
| <b>ValLevel</b>    | Overrides the outbound alphanumeric data validation table from the rule. |
| <b>ValErrLevel</b> | Overrides the outbound acceptable error level from the rule.             |
| <b>ValMap</b>      | Overrides the outbound alphanumeric data validation table from the rule. |

### EDI envelope standard generic properties

The EDI envelope standard generic properties enable you to get or set information in the EDI envelope standard segments. They have generic names that can apply to any of the EDI standard types. For example, `IchgSndrId` is used for the interchange sender ID, regardless of the EDI standard type.

These properties are available when the source document is received within an EDI envelope standard. The properties can be obtained using the `GetProperty` function. If you request a property that is not present, an empty string is returned.

These values can also be set for the target EDI document using the `SetProperty` command. If set, they will override the values specified in the envelope profile.

Here is a list of the EDI envelope standard generic properties.

|                     |                                                |
|---------------------|------------------------------------------------|
| <b>IchgCtlNum</b>   | Interchange control number                     |
| <b>IchgSndrId</b>   | Interchange sender ID                          |
| <b>IchgRcvrId</b>   | Interchange receiver ID                        |
| <b>IchgSndrQl</b>   | Interchange sender qualifier                   |
| <b>IchgRcvrQl</b>   | Interchange receiver qualifier                 |
| <b>IchgDate</b>     | Interchange date                               |
| <b>IchgTime</b>     | Interchange time                               |
| <b>IchgPswd</b>     | Interchange password                           |
| <b>IchgUsgInd</b>   | Interchange usage indicator                    |
| <b>IchgAppRef</b>   | Interchange application reference              |
| <b>IchgVerRel</b>   | Interchange version/release                    |
| <b>IchgGrpCnt</b>   | Number of groups in interchange                |
| <b>IchgCtlTotal</b> | Control total from interchange trailer segment |
| <b>IchgTrxCnt</b>   | Number of transactions in interchange          |
| <b>GrpCtlNum</b>    | Group control number                           |

## Message properties

|                     |                                       |
|---------------------|---------------------------------------|
| <b>GrpFuncGrpId</b> | Functional group ID                   |
| <b>GrpAppSndrId</b> | Group application sender ID           |
| <b>GrpAppRcvrId</b> | Group application receiver ID         |
| <b>GrpDate</b>      | Group date                            |
| <b>GrpTime</b>      | Group time                            |
| <b>GrpPswd</b>      | Group password                        |
| <b>GrpVer</b>       | Group version                         |
| <b>GrpRel</b>       | Group release                         |
| <b>GrpTrxCnt</b>    | Number of transactions in group       |
| <b>TrxCtlNum</b>    | Transaction control number            |
| <b>TrxCode</b>      | Transaction code                      |
| <b>TrxVer</b>       | Transaction version                   |
| <b>TrxRel</b>       | Transaction release                   |
| <b>TrxSegCnt</b>    | Number of segments in the transaction |
| <b>BegEnv</b>       | Begin Envelope                        |
| <b>EndEnv</b>       | End Envelope                          |
| <b>BegGrp</b>       | Begin Group                           |
| <b>EndGrp</b>       | End Group                             |

## EDI envelope standard specific properties

The EDI envelope standard specific Properties also enable you to get or set information in the EDI envelope standard segments. These have names that are specific to a particular EDI standard, and specify a particular segment and element number. For example, ISA04 is used for the fourth element of the ISA segment in an X12 envelope.

All of these properties are used to obtain the value of common data contained in an EDI envelope standard segment. These properties are available when the source document is received within an EDI envelope standard. These properties can be obtained using the `GetProperty` function. If you request a property that is not present, an empty string is returned.

These values can also be set for the target EDI document using the `SetProperty` command. If set, they will override the values specified in the envelope profile.

Here is a list of the EDI envelope standard specific properties.

|              |                                                                                                                                                |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ISAnn</b> | “nn” is 01 through 16. Use attribute ISA01 through ISA16 to obtain information from each element in the ISA segment contained in the envelope. |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------|

|                |                                                                                                                                                                                                                                                                                                                                                                  |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>GSnn</b>    | “nn” is 01 through 08. Use attribute GS01 through GS08 to obtain information from each element in the GS segment contained in the envelope.                                                                                                                                                                                                                      |
| <b>STnn</b>    | “nn” is 01 or 02. Use attribute ST01 and ST02 to obtain information from each element in the ST segment contained in the envelope.                                                                                                                                                                                                                               |
| <b>SEnn</b>    | “nn” is 01 through 02. Use attribute SE01 and SE02 to obtain information from each element in the SE segment contained in the envelope.                                                                                                                                                                                                                          |
| <b>GEnn</b>    | “nn” is 01 through 02. Use attribute GE01 and GE02 to obtain information from each element in the GE segment contained in the envelope.                                                                                                                                                                                                                          |
| <b>IEAnn</b>   | “nn” is 01 through 02. Use attribute IEA01 and IEA02 to obtain information from each element in the IEA segment contained in the envelope.                                                                                                                                                                                                                       |
| <b>UNBeess</b> | “ee” is 01 through 11. This is the position of either the element or composite in the segment. “ss” is 01 through 04. This is optional. If “ee” is a composite element, then “ss” is the position of the element within the composite. Use attribute UNB0101 through UNB11 to obtain information from each element in the UNB segment contained in the envelope. |
| <b>UNGeess</b> | “nn” is 01 through 08. This is the position of either the element or composite in the segment. “ss” is 01 through 04. This is optional. If “ee” is a composite element, then “ss” is the position of the element within the composite. Use attribute UNG01 through UNG08 to obtain information from each element in the UNG segment contained in the envelope.   |
| <b>UNHeess</b> | “nn” is 01 through 07. This is the position of either the element or composite in the segment. “ss” is 01 through 04. This is optional. If “ee” is a composite element, then “ss” is the position of the element within the composite. Use attribute UNG01 through UNG07 to obtain information from each element in the UNG segment contained in the envelope.   |
| <b>UNTnn</b>   | “nn” is 01 through 02. Use attribute UNT01 and UNT02 to obtain information from each element in the UNT segment contained in the envelope.                                                                                                                                                                                                                       |
| <b>UNEnn</b>   | “nn” is 01 through 02. Use attribute UNE01 and UNE02 to obtain information from each element in the UNE segment contained in the envelope.                                                                                                                                                                                                                       |
| <b>UNZnn</b>   | “nn” is 01 through 02. Use attribute UNZ01 and UNZ02 to obtain information from each element in the UNZ segment contained in the envelope.                                                                                                                                                                                                                       |
| <b>BGnn</b>    | “nn” is 01 through 07. Use attribute BG01 through BG07 to obtain information from each element in the BG segment contained in the envelope.                                                                                                                                                                                                                      |

## Message properties

### EGnn

“nn” is 01 through 04. Use attribute EG01 through EG04 to obtain information from each element in the EG segment contained in the envelope.



---

## Part 3. Send and Receive Maps



---

## Chapter 13. Send and Receive mapping

Send maps and Receive maps are the original map types in WebSphere Data Interchange. These maps are primarily used for migration and compatibility with maps from earlier versions of WebSphere Data Interchange.

Send maps contain a set of mapping instructions that describe how to translate a proprietary application data document into an EDI standard transaction. Receive maps contain a set of mapping instructions that describe how to translate an EDI standard transaction into a proprietary application data document.

---

### Creating a send or Receive map

You can create a new send or Receive map only after you create a data format. In some cases, you might need to create a new map to meet the data requirements of a particular trading partner. You can also create a new map from an existing data format to meet new requirements. The EDI standard transaction you are going to work with must also exist in WebSphere Data Interchange. For information about creating data formats, see Chapter 4, “Data formats,” on page 25.

The following procedure shows how to use WebSphere Data Interchange to make basic associations between fields and data elements.

1. In the Map list window, select either the **Send Maps** or **Receive Maps** tab.
2. Click New on the tool bar.  
The Create map editor opens.
3. Type a map name in the Map Name field.

**Notes:**

- a. You can use both letters and numbers to identify your map.
- b. Letters display in capitals.
- c. You cannot type spaces within the name.

To check the names of existing maps, click Show Existing Map Names.

4. Enter a description of the map in the Description field. This field is optional.
5. Click Next to continue.  
The Source Dictionary list is displayed.
  - For Send maps, this is a list of data format dictionaries.
  - For Receive maps, this is a list of EDI standard dictionaries.
6. Select the dictionary for the source document.
7. Click Next. A list of the available document definitions within the selected dictionary is displayed.
  - For Send maps, this is a list of data formats within the selected data format dictionary.
  - For Receive maps, this is a list of all transactions within the selected EDI standard dictionary.

## Creating a send or Receive map

8. Select the document definition to be used as the target document definition in your map, and click Next.  
The appropriate opens.
  - For Send maps, this page contains a list of EDI standard dictionaries.
  - For Receive maps, this is a list of data format dictionaries.
9. A list of the available document definitions within the selected dictionary is displayed.
  - For Send maps, this is a list of all transactions within the selected EDI standard dictionary.
  - For Receive maps, this is a list of data formats within the selected data format dictionary.
10. Select the definition and click next.
  - For Send maps, the Confirmation window opens. Proceed to step 12.
  - For receive mps the Compliance Checking wizard opens. This page offers you the choice of checking all EDI Standard Data for Compliance to the EDI standard, or only checking mapped data. Proceed to step 11.
11. Select the compliance checking you want on this map, then click Next.  
The Confirmation window opens.
12. Confirm the selections displayed. If correct, click Finish to save the information.  
After saving the information, the Map editor opens with the **Details** tab in front. The **Details** tab is where you can drag components of your data format and drop them on data elements within the transaction. For more information about the how to drag-and-drop components in the Map editor, see “General editing procedures” on page 95
13. You can drag fields onto data elements until you have mapped all of the information required for this map.
14. Enter any comments on the map in the Comments tab.
15. When you have completed mapping, click Save on the tool bar to save the map.

---

## Using the mapping data element editor

The Mapping Data Element editor enables you to use the advanced mapping capabilities of WebSphere Data Interchange with data elements and fields. You can:

- Use an accumulator to store, count, and add values to a field or data element. Accumulators are used for such actions as counting segments in a transaction for later placement in a transaction’s trailer record. For more information about accumulators, see “Using accumulators” on page 241.
- Associate a WebSphere Data Interchange literal or other mapping commands with a field or data element. Literals are used for such actions as providing data to trading partners that your application does not contain. For more information about literals, see “Using literals and mapping commands” on page 211.
- Use any of special handling options in WebSphere Data Interchange on a field or data element. Special handling options are used for such actions as editing dates, verifying data in a field against predefined lists, and converting data from one value to another.

## Applying advanced mapping capabilities to a field or data element

1. Double-click a data element in an EDI transaction that has not been mapped, or if the element has been mapped, then double-click of the mappings below it.

**Note:** If this is a Receive map, the Qualified Element Support window opens. Select Qualified Element Support or normal support, as appropriate. If this is a Send map, go to step 3.

2. Click Normal.  
The Mapping Data Element editor opens.
3. Select the mapping capability you want to apply to this field or data element. You can use more than one.
  - To use a literal, type any literal or mapping commands you desire to use on the field or data element in the Literal or Mapping Commands field. See “Using literals and mapping commands” on page 211 for more information.
  - To use an accumulator, select an accumulator from the Accumulators list. For each accumulator, you must select an action from the Actions list. You can use up to four accumulators on any data element or field. See “Using accumulators” on page 241 for more information.
  - To use a special handling option on a field or data element, click Special Handling. See “Using the special handling button” for more information.
  - To create another element mapping for this data element or field, click Repeat. See “Using the Repeat Button” on page 209 for more information.
4. If you wish, you can view attributes of the data element and field you are mapping.
  - To view the attributes of the data element on which you are working, click Element Attributes.  
The Data Element Attributes window opens. You see the same information about the data element that is displayed on the **General** tab of the Data Elements editor.
  - To view the attributes of the field on which you are working, click Field Attributes.  
The Field Attributes window opens. You see the same information about the field that is displayed on the **General** tab of the Data Format Field editor.
5. Type any comments on the mapping in the Comments field.
6. Click OK to save the mapping.

## Using the special handling button

Use the Special Handling button when you want to request special handling on data elements and fields for both send and Receive maps. You can:

- Change date formats from one format to another.
- Verify field values against predefined lists.
- Translate field values from one predefined value to another.
- Call an external program to perform custom processing on the value contained in a field or data element.

## Mapping data element editor

- Specify the length and position of fields when mapping a concatenation for Send maps or specify the length and position of fields that are included in a substring for Receive maps.

To request special handling options:

1. While working on a data element or field in the Mapping Data Element editor, click Special Handling.

The Data Element Special Handling window opens.

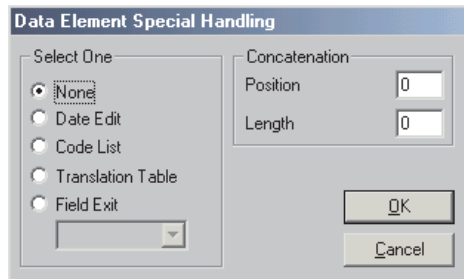


Figure 11. Data Element Special Handling window

2. Click the option corresponding to the Special Handling option you desire.

Table 26. Special Handling Options

| Option    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| None      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Date Edit | <p>Use this option to convert a date from one format to another. In a Send Map, the format of the date in the data is specified in the list. It will be converted to the format set in the EDI Standards. In a Receive Map, the date in the Data Element is converted to the format specified by the value selected in the list.</p> <p>Whenever your application requires a date in a format that is different from the EDI Standard, use the appropriate date edit code. For Send Maps, all date formats are changed to yymmdd. For Receive Maps, all date formats are changed from yymmdd. If the predefined date edits do not satisfy your date edit requirements, you can use the any-to-any date conversion facility described in “Date conversion special operators” on page 206.</p> |
| Code List | <p>Use the Code List option to verify values that appear in the data against a pre-defined Code List. For instance, if the Data Format Field or Data Element can contain only a certain range of values, you can enter those values into a Code List. When WebSphere Data Interchange processes the Data Element, it references the Code List and checks the value of the data against it. If the data contains a value that does not appear in the Code List, WebSphere Data Interchange returns a processing error.</p> <p><b>Note:</b> You must create a Code List or load it from an EDI Standard before it appears in the list.</p>                                                                                                                                                     |

Table 26. Special Handling Options (continued)

| Option                                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Translation Table                       | <p>Use the Translation Table option to convert the value that appears in the Data Format Field or Data Element into another value. For instance, you can use this option to translate part numbers used for your parts by a trading partner into the values you use in your system. When WebSphere Data Interchange processes the Data Element, it will reference the Translation Table selected in this mapping and converts the value as defined in the table. Translation Tables can also be used to translate values between those used by your application and those used in the EDI Standard.</p> <p><b>Note:</b> Both Forward Translation Tables and Reverse Translation Tables are included in the list.</p>                                                                          |
| Field Exit                              | <p>Use this option to call a program outside WebSphere Data Interchange to perform additional processing on the value contained in the Data Format Fields or Data Element.</p> <p><b>Note:</b> You must create a User Exit profile for your field exit routine before it can appear in the list. For more information about User Exit profile, see “User Exit profile” on page 208.</p>                                                                                                                                                                                                                                                                                                                                                                                                       |
| Concatenation Position (Send maps only) | <p>For Send maps, use this field to put a part of the data into the Data Element. This field specifies the position to begin concatenation in the characters to be included in the new Data Element. Use the Concatenation Length field to specify how many characters are included. The concatenation option enables you to select two or more Data Format Fields in a business document you are sending and combine parts of each Field into a single Data Element. Note: If you map two fields to the same Data Element, those fields concatenate automatically. Leading zeros and trailing blanks are stripped out. This option is only available for Send Maps when no other options are selected on the Data Element Special Handling dialog. It is not available for Receive Maps.</p> |
| Concatenation Length (Send maps only)   | <p>Use this field to specify the number of characters to move to the new Data Element. This field is used with the Concatenation Position field. This option is only available for Send Maps when no other options are selected on the Data Element Special Handling dialog. It is not available for Receive Maps</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Substring Position (Receive maps only)  | <p>Use this field to extract part of the data from the Data Element. This field specifies the position to begin extracting the characters from the Data Element. Use the Substring Length field to specify how many characters are extracted. The substring option enables you to weed out unnecessary data received from a trading partner. For example, if the first two bytes of a given Data Element are always the same, you can instruct WebSphere Data Interchange not to include those characters in your Data Format Field. This option is only available for Receive Maps when no other options are selected on the Data Element Special Handling dialog. It is not available for Send Maps.</p>                                                                                    |

Table 26. Special Handling Options (continued)

| Option                               | Description                                                                                                                                                                                                                                                                                        |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Substring Length (Receive maps only) | Use this field to specify the number of characters to include in the data. This field is used with the Substring Position field. This option is only available for Receive Maps when no other options are selected on the Data Element Special Handling dialog. It is not available for Send Maps. |

3. Select the item you need from the list.
4. Click OK to save your options.

**Note:** You can use more than one Special Handling option on any field by using repeat mapping. Click Repeat to create another instance of the element mapping you are working with. Edit that instance with the Mapping Data Element editor using the Special Handling button again to select another option.

### Date conversion special operators

WebSphere Data Interchange can perform any-to-any date conversions. The format of the any-to-any date conversion operator is:

&E(variable FD mask TD mask)

Where:

**variable**

The value to be converted.

**FD** From Date operator which signals that the following variable establishes the mask that describes the date within variable.

**TD** To Date operator which signals that the following variable establishes the mask that describes the date that is wanted.

**mask** The mask that describes the FROM or TO date. A mask consists of the symbols which identify the date provided or date wanted. Symbols can be in upper or lower case. Any value in the mask that is not one of the symbols listed is expected to be physically part of the source data (FD) or will become physically part of the result data (TD). The valid symbols are:

**CC** Century.

**YY** Year.

**MM** Month of year.

**DD** Day of month.

**D** Day of month as a single character, if possible.

**HH** Hour of day.

**MM** Minute of hour.

**II** Minute of hour.



MM can be used when it immediately follows HH as in HHMM; however, if you want minute followed by hour, you must use IIHH, because MMHH would be interpreted as month of year and hour of day.

**SS** Second of minute

**WW** Week of year (1 through 52)

**K** Day of week (Monday=1, Tuesday=2, and so on)

D can be used if it immediately follows WW as in WWD; however, if you want day of week followed by week, you must use KWW, because DWW would be interpreted to be day of month and week of year.

**JJJ** Julian day of year

**Q** Quarter (1,2,3,4)

**E** Semester (1,2)

**ZZZ** Time zone

**TM** Textual month (for example, January, February)

TM can be followed by the name of a translate table to convert a textual month to a numeric month (FD), or from a numeric month to a textual month (FD). If a table name is not provided, the default table names are DIMONTXT to translate from text to numeric and DIMONNUM to translate from numeric to text. A table name is indicated using parenthesis, for example; TM(tablename), where tablename must be a constant.

Processing considerations for WebSphere Data Interchange after creating the From Date and before creating the To Date:

- SS (seconds) default is 0.
- CC (century) default is 19 when the YY (year) is greater than 10 and to 20 otherwise.
- JJJ (Julian day) is created based on WW (week of year) and K (day of week) if not otherwise provided and WW and K were provided.
- JJJ (Julian day) is created based on MM (month of year) and DD (day of month) if not otherwise provided and MM and DD were provided.
- JJJ (Julian day) is used to determine WW (week of year), K (day of week), if either WW or K was not provided.
- JJJ (Julian day) is used to determine MM (month of year) and DD (day of month), if either MM or DD was not provided.
- Q (Quarter) is determined based on MM (month of year), if not otherwise provided.
- E (Semester) is determined based on MM (month of year), if not otherwise provided.

The following are examples using CONSTANTS for all values:

- Simple example to remove delimiters &E('96/06/07' FD 'YY/MM/DD' TD 'YYMMDD') yields '960607'.

## Mapping data element editor

- Simple example to remove delimiters and rearrange `&E('96/06/07' FD 'YY/MM/DD' TD 'MMDDYY')` yields '060796' and `&E('96/06/07 EDT' FD 'YY/MM/DD ZZZ' TD 'ZZZ MMDDYY')` yields 'EDT 060796'.
- Change delimiters and convert from one form to another `&E('96/06/07' FD 'YY/MM/DD' TD 'YY:JJJ')` yields '96:157'
- Textual month to Numeric month `&E('June 7, 1996' FD 'TM D, CCYY' TD 'YYMMDD')` yields '960607'
- Numeric month to Textual month `&E('060796' FD 'MMDDYY' TD 'TM D, CCYY')` yields 'June 7, 1996'
- Numeric month to Textual month with a special translate table that uses abbreviations for the months `&E('060796' FD 'MMDDYY' TD 'DDTM(ABBREV)CCYY')` yields '07JUN1996'

### User Exit profile

A User Exit profile defines a user provided program or exit routine to WebSphere Data Interchange. It provides WebSphere Data Interchange with the information it needs to call user provided programs and exit routines. WebSphere Data Interchange can call these at various stages of translation. The name of a User Exit profile is used as a logical name for the user provided program or exit routine throughout WebSphere Data Interchange. A User Exit profile identifies the name of the program or exit routine load module. It also specifies the programming language the program or exit routine is compiled as. The programming language is necessary so WebSphere Data Interchange can provide the correct linkage to the program or routine.

You need a User Exit profile defined for each user provided program or exit routine you use. WebSphere Data Interchange supports programs for:

- Network communications
- Security
- Data Mapping

There are two types of user exits supported by WebSphere Data Interchange. One type defines user exits used in Data Transformation. These are known as Data Transformation Exits . These are currently used in Data Transformation Maps, Validation Maps and Functional Acknowledgement Maps. The other exit type is known as Send or Receive Exits. Send or Receive exits are used in Send Maps, Receive Maps, network communications, network security, pre-translation, and post-translation. See Data Transformation field exit routine for additional information about Data Transformation Exits. See the *WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide*, SC34-6217-01 for additional information about other exit types.

Send or Receive User Exit profiles identify what functions each user exit can support. See the *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01 for information about creating User Exit profiles.

## Using the Repeat Button

Use the Repeat button when you want to duplicate the element mapping you completed in the Mapping Data Element editor. One reason to duplicate an element mapping is to apply more than one special handling option for the data element or field. For example, if you want to use more than one literal command on a field, you need to repeat the element mapping as you can only use one literal command per element mapping.

You can also use repeat mapping to add data to your application when it does not occur in the EDI standard and vice versa. The following example shows how you can add data to your application that does not occur in the EDI standard through a Receive map using repeat mapping.

Say that an EDI transaction contains a purchase order total in a particular data element in a particular segment. A purchase order total less than \$100 is an error that you want to report. You can use the Repeat button to repeat the element mapping and enter the mapping commands necessary to accomplish this task, as follows.

To repeat an element mapping:

1. Double-click the data element that contains the purchase order total.  
The Mapping Data Element editor opens.
2. Enter in the Literal or Mapping Command field the mapping command &SAVE TOTPO to save the data element value in variable TOTPO.
3. Click Repeat.  
The element mapping is saved and another element mapping is created. The Mapping Data Element editor redisplayes with the new element mapping. The saved element mapping appears under the associated data element in the Map editor.  
You can select any options you desire for the new element mapping.
4. Enter the mapping command in the Literal or Mapping Command field to check if the value saved is less than 100 and to issue a user error.  
&IF (TOTPO < 100) &ERR(1,1,0,"PO is less than \$100.00").
5. Click OK and finish editing the map.  
The new element mapping is designated Literal of (the entered mapping command).

## Setting an application control key

To make it easier to find a series of documents in WebSphere Data Interchange's Document Store database, you can set up keys in your maps using application control fields. The application control field contains an ID used to identify the document in the Document Store, such as the purchase order number in a purchase order document. In the Application Control Fields window, you can select up to eight fields, literals and variables to used to construct the application control field.

For example, say your company has a string of characters that occur in front of a purchase order number. Your trading partners likely send only the number, but you can use the Application Control Fields window to have WebSphere Data Interchange add the characters you desire to their values. You can also use WebSphere Data Interchange variables to modify data in the application control field.

## Application control key

To make documents received from or sent to a particular trading partner easier to find in the Document Store, you can add a string of characters representing that company to the application control field. That character string becomes something you can search on to build a list of documents when viewing the Document Store.

For more information about the Document Store, see the *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

**Note:** Application control fields are optional; the translator does not require them.

To set up application control fields:

1. Click the **Details** tab of the map for which you want to set up application control fields.
2. On the data format (left) side of the page, click the field you want to set as an application control field and hold down the left mouse button. Drag the field over the Application Control Fields pane of the transaction (right) side of the page, and release the mouse button.

The Application Control Fields window opens.

The Path column contains the name of the data format loop(s), record and (possibly) structures in which the field you dragged occurs. The Field column contains the name of the field. The Length column contains the length of the field.

You can add as many as eight fields to the window.

3. If you want to add fixed data to the application control fields, click Add &LIT.

The Application Control Fields - &LIT Support window opens.

- a. Type in the Enter Literal field the data you wish to add to the application control field.
- b. Type in the Enter Length field the length of the data you wish to add to the application control field.
- c. Click OK.

The data you typed is displayed in the Field column in a row below the field to which you added it. The data is preceded by &LIT.

4. If you want to include the data from a WebSphere Data Interchange variable, click Add &VAR.

The Application Control Fields - &VAR Support window opens.

- a. Type in the Enter Variable field the name of the WebSphere Data Interchange variable you wish to use.
- b. Type in the Enter Length field the length of the data you wish to add to the application control field.
- c. Click OK.

5. When you have completed your setup, click OK.

The **Details** tab of the Map editor redisplay.

---

## Using literals and mapping commands

WebSphere Data Interchange literals and mapping commands let you add data to a transaction that is not contained in your business application for Send maps, or put data into your application data that is not contained in the transaction. When your application does not have specific information required by the transaction, or when you need to pass specific information, you can map a literal to the data element. Conversely, when your application requires data that is not specified in the transaction, you can supply the information by mapping a literal.

A literal is a value that you specify for the field or data element. The value can be a constant or it can be calculated by any of several WebSphere Data Interchange expressions.

A mapping command is a WebSphere Data Interchange command that begins with an ampersand (&). For a list of mapping commands, search for the key mapping commands in WebSphere Data Interchange Client Help.

### Adding a literal or mapping command to a map

You can add literals or mapping commands to maps through the Mapping Data Element editor, as follows.

1. Double-click an element mapping that has been mapped to a field from a data format, or double-click a data element that has not been mapped. If this is a Receive mapping, the application data field is required for a literal.  
The Mapping Data Element editor opens.
2. Type the name of the literal and any required expressions. Note that literals are case-sensitive. Mapping commands and variable names are not.
3. Complete any other mapping you require from the Mapping Data Element editor, and click OK.

### Literals and data types

The following apply to literals used in both send and Receive maps:

- For data types BN, Bn, HX, IT, In, Ln, PD, Pn, Zd, and Zn, the translator converts the literal before placing it in the outgoing EDI standard data or the incoming application data. In Send maps, it converts the literal to character data. In Receive maps, it converts the literal to the application data type. For example, if the data type is BN, the translator converts the literal to binary and then moves it to the application field.
- Do not type a decimal point when one is implied. To use a default value of 9.99 for a field defined as P2 (packed number with two implied decimal positions), enter the literal as 999.
- Literal values for hexadecimal fields are hexadecimal strings. For example, if the application field is defined as a one-byte hexadecimal field and you want to use a default value of X'FF', enter FF as the value of the literal. For incoming data, the translator converts each two bytes of literal value to a single byte of application data.

**Attention:** Type hexadecimal numbers using their EBCDIC values, not their ASCII values.

## Literals and data types

- Specify a literal value of zero to move this value into a data element. WebSphere Data Interchange generally removes leading zeros from application data so that a field containing all blanks or all zeros will result in no value for the data element. A value of zero is treated the same as all other literal values when determining if a segment is created. The &ZEROSIG special literal can also be used to indicate that zeros within the application field are significant.
- In translation and validation tables, enter numeric values left-justified and formatted according to the data format. For example, if the data format defines a field as R2, enter the value 7 as 7.00 or the value 7.1 as 7.10.

**Note:** Before creating a new map for a specific trading partner, check to see if you can use WebSphere Data Interchange's advanced mapping functions to meet a trading partner's special processing requirements. You can set up a map so that WebSphere Data Interchange meets a trading partner's custom data and processing requirements. For more information, "Using the mapping data element editor" on page 202

---

## Translation tables

When WebSphere Data Interchange translates data from your data format to an EDI transaction or from an EDI transaction to your data format, it can also substitute one value for another. WebSphere Data Interchange substitutes values through translation tables. Use translation tables to handle:

- Differences between your data and your trading partners' data. For example, say your trading partner uses its own numbers for parts you sell. You can set up a translation table to convert the part numbers your trading partner uses to those you use. An example of such a translation table is illustrated in Table 27.

*Table 27. Translation table, differences in data*

| Local Value | Trading Partner Value |
|-------------|-----------------------|
| GLF8088     | FR0100                |
| GLF8588     | FR0600                |
| GLF8788     | FR0800                |

- Conflicts between application data and EDI standards. For example, your application uses a code for a unit of measure that does not occur in the EDI standard. You can create a translation table to substitute an EDI standard code for your code on the send side and your code for the EDI standard code on the receive side. An example of such a translation table is illustrated in Table 28.

*Table 28. Translation table, conflicts with standards*

| Local Value | Trading Partner Value |
|-------------|-----------------------|
| Boxes       | BX                    |
| Cases       | CS                    |
| Doz         | DZ                    |
| Each        | EA                    |

You associate translation tables with maps through the Special Handling button on the Mapping Data Element editor, as described on “Using the mapping data element editor” on page 202. This section describes how to set up translation tables.

For more information about Translation Tables, see the *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

WebSphere Data Interchange provides two types of translation tables:

- Forward translation tables
- “Reverse translation tables”

### Forward translation tables

The most commonly used is the forward translation table. Set up a forward translation table when you want to translate values from your application into values required by your trading partner when sending data, or translate values from your trading partner data into values required by your application when receiving data. This type of translation table contains values with a one-to-one relationship or many application values to one EDI standard value. The local value side of the table definition must be unique, as illustrated in Table 29.

*Table 29. Sample forward translation, table values*

| Target (Application Value) | Source (Standard Value) |
|----------------------------|-------------------------|
| 01                         | AA                      |
| 02                         | BB                      |
| 03                         | CC                      |
| 04                         | CC                      |

This type of translation table can be used on either send or Receive maps, when both application and EDI standard values are unique. It can be used in Send maps when only the application values are unique.

### Reverse translation tables

Set up a reverse translation table when you want to translate one or more values from your trading partner to a single value in your application. This type of translation contains values with a one-to-one relationship or many EDI standard values to one application value. The EDI standard value side of the table definition must be unique, as illustrated in Table 30.

*Table 30. Sample reverse translation, table values*

| Application Value | Standard Value |
|-------------------|----------------|
| 01                | AA             |
| 01                | BB             |
| 01                | CC             |

## Translation tables

Table 30. Sample reverse translation, table values (continued)

| Application Value | Standard Value |
|-------------------|----------------|
| 22                | DD             |
| 22                | EE             |

The procedures for creating Forward Translation tables and Reverse Translation tables are exactly the same.

### Creating a new translation table

Create a new translation table when you need to substitute values supported by your application for values supported by the trading partner's. You can also need to create a translation table to substitute values supported by your application and EDI standard transactions.

In the following procedure:

- If you are creating a Forward Translation Table, first group refers to the Source Variable group box.
  - If you are creating a Reverse Translation Table, first group refers to the EDI Standards Value group box.
  - If you are creating a Forward Translation Table, second group refers to the Target Variable group box.
  - If you are creating a Reverse Translation Table, second group refers to the Data Format Value group box.
1. In the Mapping list window, click either the **Forward Translation** tab or the **Reverse Translation** tab.
  2. Click New on the tool bar.  
The **General** opens.
  3. Type in a name for the translation table.

#### Notes:

- a. This value must be unique among Forward Translation Tables, Reverse Translation Tables, and Code Lists.
  - b. The name is displayed in uppercase letters.
  - c. You cannot type spaces within the name.
  - d. The name can be up to 8 characters long and can contain alphanumeric characters and any of the special characters "!@#%&\*()\_+='<,>.>?".
4. Enter a description for the translation table. This field is optional.
  5. In the first group of the editor, select a data type from the Data Type list. This field identifies the type of data that can be contained in the values. It can be either:
    - Character—All characters.
    - Numeric—Only numbers (0–9).
  6. In the first group, specify the maximum length of the value.



7. In the second group of the editor, select a data type from the Data Type list. This field identifies the type of data that can be contained in the values. It can be either:
  - Character – All characters.
  - Numeric – Only numbers (0–9).
8. In the second group, specify the maximum length of the value.

**Note:** The combined length of the lengths from the first and second group cannot exceed 68 characters.

9. Click New to add entries to the translation table.
10. To define the entry:
  - For Forward Translation tables, enter a source value and its corresponding target value.
  - For Reverse Translation tables, enter an EDI Standards value and its corresponding Data Format value.
11. At this point you have two options:
  - To add the entry and continue adding entries, click Insert and return to step 10.
  - To add the entry and close the editor, click OK. The entry is added to the list and the editor closes.
12. Click Save on the editor tool bar to save the translation table.

---

## Specifying send and receive usages

When you have completed a map, you must associate it with a trading partner or trading partners. WebSphere Data Interchange calls those associations send usages or receive usages, or jointly usages. Send usages are also referred to as Send map usages. Receive usages are also referred to as Receive map usages.

## Applying the minimal trading partners concept

The concept of minimal trading partners attempts to reduce the amount of time spent on administrative functions of EDI. The traditional WebSphere Data Interchange was based on the idea that each trading partner would be identified to the product through a trading partner profile and a send usage or receive usage. Thus, a WebSphere Data Interchange installation with tens of thousands of trading partners would require an equal number of profiles and usages, even though the options were identical. The WebSphere Data Interchange concept of generic usages reduces the administrative impact of this model, but does not completely meet all its needs. Some installations do not need a setup for a trading partner, relying on post-translator processes to validate EDI transactions. WebSphere Data Interchange uses a combination of techniques and terminology to accommodate this minimal administrative model.

Applying the minimal trading partners concept, usages enable you to specify the same transaction mapping for several trading partners and provide specific overrides for each trading partner. This gives you the capability to use a single map for several different trading partners. Each send or receive usage instructs WebSphere Data Interchange

## Send and receive usages

Client which components of the map are used and which are overridden for that particular trading partner. See *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01 for more information.

The following procedures can be carried out from the trading partner windows as well as the mapping windows.

## Viewing usages

To view usages:

1. In the Mapping list window, click the map for which you want to view usages, or go to the editor for that map.
2. Click View Usages on the tool bar.

WebSphere Data Interchange Client runs a query that displays the Usages list window, which contains one tab. The **Send Map Usages** opens if the map is a Send map. It displays a list of Send Usages associated with the map. The **Receive Map Usages** opens if the map is a Receive map. It displays a list of Receive Usages associated with the map.

## Creating a Send map usage

Create a new send usage after you create a new send and need to associate an existing trading partner with the map. You can also create usages to associate trading partners with existing maps. Figure 12 shows the Send Map Usage editor.

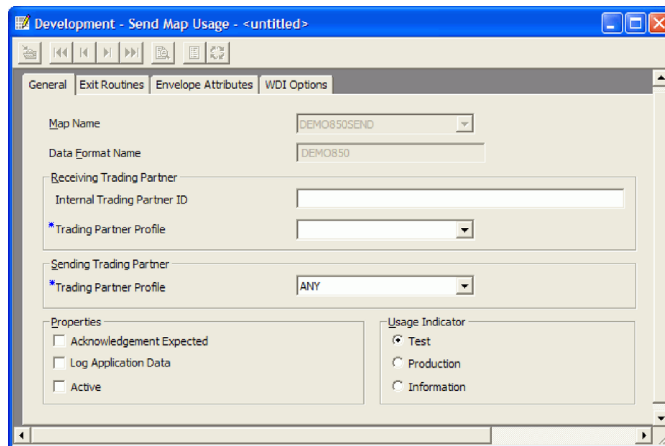


Figure 12. Send Map Usage editor

To create a Send map usage:

1. Open the Send Map Usages list window. This can be accomplished by any of the following means:
  - Select the Open Functional Area action on the File menu. This opens the Open Functional Area dialog. Select Rules and Usages and then click OK. The Rules and Usages Functional Area window will open containing list windows for Map

Rules, Send Map Usages, and Receive Map Usages. Click on the Send Map Usages tab to display the Send Map Usages list window. The list window will contain a list of all Send Map Usages selected by the current Query. Alternately you can click Rules and Usages on the navigator bar to open the Rules and Usages Functional Area window.

- Select the Open Query List action on the File menu. This will open the Query list window dialog. Select the Rules and Usages Functional Area, and then select Send Map Usages. Finally, select the Query you wish to execute and click Run. A window will open containing the Send Map Usages list window. The list window will contain a list of all Send Map Usages selected by the Query.
  - Open the Send Maps list window by clicking Mapping on the WebSphere Data Interchange Client navigation bar. Click on the Send Maps tab to display the Send Maps list window. Select the Send Map associated with the Send Map Usage you wish to create. Click Rules and Usages on the toolbar or select Rules and Usages from the Actions menu. A window will open containing the Send Map Usages list window. It will contain a list of all Send Map Usages associated with the selected Send Map.
  - Open the Trading Partners list window by clicking Trading Partner on the WebSphere Data Interchange Client navigation bar. Click on the Trading Partners tab to display the Trading Partners list window. Select the Trading Partner profile associated with the Send Map Usage you wish to create. Click Rules and Usages on the toolbar or select Rules and Usages from the Actions menu. A window will open containing the Send Map Usages list window. It will contain a list of Send Map Usages associated with the selected Trading Partner profile.
2. Click New on the list window toolbar. The Send Map Usage editor is opens with the **General** tab in front.
  3. Select the name of the map to associated with this Send Map Usage.

**Note:** When you select a map, then the Data Format Name field are filled in automatically using the map's source document. This cannot be changed. When the Send Map Usages list window is opened from the Send Maps list window or the Send Map editor, the Map Name and Data Format Name fields are filled in and cannot be changed .

4. Enter an internal trading partner ID if required.  
Begin this field with an ampersand (&) followed by blanks or a three character routing code if you are defining a generic Send Map Usage.  
This field identifies the name of trading partner whom you will send the Transaction. This is usually a vendor or customer number that your application uses to see the trading partner where the Transaction is being sent. An internal trading partner ID obtained from the source document at the beginning of the translation process can be used to help determine which Send Map Usage is used to identify the Send Map used to translate the source document. It can also help to identify the Trading Partner profile representing the receiving trading partner.
5. Select the name of the Receiving Trading Partner profile.  
If you are going to create a generic Send Map Usage, leave the receiving Trading Partner profile field blank.

## Send and receive usages

**Note:** When the Send Map Usages list window is opened from the Trading Partners list window or the Trading Partner editor, this field might be pre-filled with the name of the selected Trading Partner profile. This occurs when the selected Trading Partner profile is trading partner type EDI Trading Partner. You will not be able to change this value in this case.

6. Select the name of the Sending Trading Partner profile.

**Note:** When the Send Map Usages list window is opened from the Trading Partners list window or the Trading Partner editor, this field might be pre-filled with the name of the selected Trading Partner profile. This occurs when the selected Trading Partner profile is trading partner type Application Trading Partner. You will not be able to change this value in this case.

7. In the Usage Indicator pane of the tab, indicate whether the Send Map Usage is used when the source document is a production, test, or information document.  
The value of this field is used to set a flag in the EDI Standard Envelope when the created Transaction is enveloped. This field is also used to help determine which Send Map Usage is used to identify which Send Map is used to translate the source proprietary document. Documents being translated are classified as production, test or information documents. For Data Formats, classification is determined from a flag on the C record, when using C and D records. When using raw data format data formats, classification is production unless the RAWUSAGE keyword is present in the PERFORM command.
8. Indicate if a functional acknowledgement is expected from the trading partner. The outbound EDI document is reconciled with the inbound functional acknowledgement if the Document Store is active.  
A marked Acknowledgement Expected check box indicates that the outbound EDI data is expecting a functional acknowledgement returned from the Trading Partner. An unmarked check box indicates that a functional acknowledgement is not expected from the Trading Partner.
9. Indicate whether you want WebSphere Data Interchange to save an image of the application data in the Event Log.  
A marked Log Application Data check box indicates that generated application data is logged to the Event Log. An unmarked check box indicates that generated application data is not logged to the Event Log.
10. Indicate if the Send Map Usage is active. Inactive Send Map Usages are not used when trying to determine what Send Map is used to translate a document.  
When the internal trading partner ID is not blank, there can only be one active Send Map Usage with the same document, sending Trading Partner profile, internal trading partner ID, and usage indicator combination. When the internal trading partner ID is blank, there can only be one active Send Map Usage with the same document, sending Trading Partner profile, receiving Trading Partner profile, internal trading partner ID, and usage indicator combination.
11. Select the **Exit Routines** tab and complete the fields as necessary. See Table 31 on page 219 for descriptions of the fields on this tab.  
The **Exit Routines** tab page of the Send Map Usage editor holds information related to a post-translation exit routine that can be called. Group and Transaction level authentication and encryption keys can also be provided on this tab page is a

user written authentication or encryption routine will be used.

Table 31. Send map usages Exit Routines tab fields descriptions

| Field                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Post-Translation Exit Routine  | <p>This field can be used to indicate the WebSphere Data Interchange is supposed to call a user-written exit routine after translating the document.</p> <p>If a post-translation exit routine is called, select the name of a User Exit profile from the list. The User Exit profile identifies the exit routine that will be called. Only User Exit profiles identified as post-translation exit routines will be listed in the list.</p> <p>See the <i>WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide</i>, SC34-6217-01 for additional information about exit routines.</p> |
| Group Encryption Key           | <p>If you are using a user-written encryption routine, use this field to enter the encryption key that WebSphere Data Interchange puts in the group security Segments and passes to a user-written encryption routine.</p> <p>The field can contain up to 16 characters in any combination of A–Z, 0–9 and any of the special characters “!@#%&amp;*()_+ =';&lt;,&gt; .?/”.</p>                                                                                                                                                                                                                            |
| Group Authentication Key       | <p>If you are using a user-written authentication routine, use this field to enter the authentication key that WebSphere Data Interchange puts in the group security Segments and passes to a user-written authentication routine.</p>                                                                                                                                                                                                                                                                                                                                                                     |
| Transaction Encryption Key     | <p>If you are using a user-written encryption routine, use this field to enter the encryption key that the WebSphere Data Interchange puts in the Transaction set security Segments and passes to a user-written encryption routine.</p>                                                                                                                                                                                                                                                                                                                                                                   |
| Transaction Authentication Key | <p>If you are using a user-written authentication routine, use this field to enter the name of the authentication key that the WebSphere Data Interchange puts in the Transaction set security Segments and passes to a user-written authentication routine.</p>                                                                                                                                                                                                                                                                                                                                           |

12. Select the **Envelope Attributes** tab and complete the fields as necessary. See Table 32 on page 220 for descriptions of the fields on this tab.

The **Envelope Attributes** tab holds information related to EDI Standard Envelopes. Some of the information about this tab is used to override the default EDI Standard Envelope that normally would be generated for a target EDI Standard Transaction. Typically, you want to provide the name of an Envelope Profile

## Send and receive usages

Table 32. Send map usages Envelope Attributes tab field descriptions

| Field         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Envelope Type | <p>This field is used to identify the type of EDI Standard Envelope that will be generated for the target EDI Standard Transaction when an envelope is created for it. Valid values are:</p> <ul style="list-style-type: none"><li>• UN/EDIFACT</li><li>• ICS - Interchange Control Structures</li><li>• UN/TDI - Trade Data Interchange</li><li>• UCS - Uniform Communication Standard</li><li>• X12</li><li>• Dictionary Created from an XML DTD</li><li>• No Interchange Envelope Will Be Used</li><li>• Default Envelope</li></ul> <p>EDI Standard Transactions that are a part of the ICS, UCS, and X12 EDI standards can be sent without an interchange envelope.</p> <p>A blank value is equivalent to the Default Envelope value.</p> <p>If Default Envelope is selected, the envelope type identified by the EDI Standard Dictionary associated with the EDI Standard Transaction is used. A value other than Default Envelope overrides the value in the EDI Standard Dictionary.</p> <p>Dictionary Created from an XML DTD is used to identify EDI Standard Dictionaries that were created from an XML DTD. This is a function used to support XML to Data Format translation using Send Maps and Receive Maps.</p> |

Table 32. Send map usages Envelope Attributes tab field descriptions (continued)

| Field                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Envelope Profile Name   | <p>This field contains the name of the Envelope Profile that is used to provide some values when the EDI Standard Envelope is generated for the target EDI Standard Transaction.</p> <p>The values that appear in this list are based upon the value contained the Envelope Type field. When the Envelope Type field indicates that a UN/EDIFACT envelope will be used, this field will list all E Envelope profiles. I Envelope profiles are listed when the Envelope Type field indicates that an ICS envelope will be used. When the Envelope Type field specifies that a UN/TDI envelope will be used, this field will list all T Envelope profiles. When the Envelope Type field indicates that a UCS envelope will be used, this field will list all U Envelope profiles. All X Envelope profiles will be listed when the Envelope Type field indicates that an X12 envelope will be used. This field is disabled when the Envelope Type field indicates that the default envelope or the envelope will not include the interchange portion of the envelope.</p> <p>Select the Envelope profile to use when creating an Envelope related to the target EDI Standard Transaction. If the field is left blank, the Envelope profile with the same name as the EDI Standard Dictionary associated with the target EDI Standard Transaction will be used.</p> <p>A generic Envelope profile name can be specified in this field. You can specify a generic Envelope profile name by entering an ampersand (&amp;) followed by a 1 to 6 character base name. During translation, the Envelope profile suffix from the Trading Partner profile will be appended to the base name to dynamically determine the name of the Envelope profile to use when enveloping the document.</p> |
| Application Sender ID   | <p>The sender ID identifies the specific sender of the document, such as a department number.</p> <p>The value specified in this field is used as the sender ID in the functional group header of the EDI Standard Envelope that might be generated for the target EDI Standard Transaction. The sender ID maps to the EDI Standard Data Element with data type AS. A value entered here overrides the sender ID specified in the Envelope profile. Providing an application sender ID in the C record (when using C and D records) overrides the sender ID in the Envelope profile and any value specified in this field.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Application Receiver ID | <p>The receiver ID identifies the specific receiver of the document, such as a department number.</p> <p>The value specified in this field is used as the receiver ID in the functional group header of the EDI Standard Envelope that might be generated for the target EDI Standard Transaction. The receiver ID maps to the EDI Standard Data Element with data type AR. A value entered here overrides the receiver ID specified in the Envelope profile. Providing an application receiver ID in the C record (when using C and D records) overrides the receiver ID in the Envelope profile and any value specified in this field.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## Send and receive usages

Table 32. Send map usages Envelope Attributes tab field descriptions (continued)

| Field                | Description                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Application Password | This field is used as the password in the functional group header of the EDI Standard Envelope that might be generated. The password maps to the EDI Standard Data Element with data type PW. A value entered here overrides the password specified in the Envelope profile.                                                                                                                                                 |
| Group Security       | Select the name of the Network Security profile that contains the information needed by WebSphere Data Interchange at the functional group level to: <ul style="list-style-type: none"> <li>• Build security segments.</li> <li>• Call user-written exit routines for security and data compression.</li> </ul> <p>A value entered here overrides the Network Security Profile specified in the Trading Partner profile.</p> |
| Transaction Security | Select the name of the Network Security profile that contains the information needed by WebSphere Data Interchange at the Transaction set level to: <ul style="list-style-type: none"> <li>• Build security Segments.</li> <li>• Call user-written exit routines for security and data compression.</li> </ul> <p>A value entered here overrides the Network Security Profile specified in the Trading Partner profile.</p>  |

13. Select the **WDI Options** tab. Identify any processing options that are used when processing a document using this Send Map Usage. See Table 33 for descriptions of the fields on this tab.

Table 33. Send map usages WDI options tab field descriptions

| Fields           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Validation Level | Use this field to indicate the level of automatic validation you would like performed during the translation. The valid values are: <p><b>Value      Meaning</b></p> <p><b>Only Mandatory Validation</b><br/>No additional validation over the mandatory validation done by WebSphere Data Interchange to convert data between the source data type and the EDI Standard data type. With this level of validation, any Code Lists specified in the Send Map or in an EDI Standard Data Element are ignored.</p> <p><b>Mandatory Validation Plus Code Lists</b><br/>Includes mandatory validation plus the use of the Code Lists specified in the Send Map and in EDI Standard Data Elements.</p> <p><b>Mandatory Validation Plus Code Lists Plus Data Type Verification</b><br/>Includes mandatory validation plus the use of the Code Lists specified in the Receive Map and in EDI Standard Data Elements. In addition, verification that the data values supplied are consistent with the Data Element's data type. For instance, use this level of validation to verify that a Data Element that is supposed to contain a date has a valid date, or a numeric Data Element contains numeric data only.</p> |



Table 33. Send map usages WDI options tab field descriptions (continued)

| Fields                             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Acceptable Error Level             | <p>Use this field to indicate what level of error will be acceptable for a translation to be considered successful. Valid values are:</p> <ul style="list-style-type: none"> <li>• Allow only translations with no errors.</li> <li>• Allow a translation that has no error or has only Data Element errors.</li> <li>• Allow a translation that has no error, has Segment errors, or has Data Element errors.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Alphanumeric Data Validation Table | <p>This field is used to specify the name of an alternate alphanumeric data validation table that is to be used for validating alphanumeric character data during translation. The table identified in this field contains a subset of the characters found in the character data validation table. If an alphanumeric data validation table is specified in this field, it will be used for the translation instead of the default alphanumeric data validation table. When this field is blank, the default alphanumeric data validation table will be used.</p> <p>The alphanumeric data validation table is defined as a Code List in the EDI Standards Functional Area. ALPHANUM is the name of the default table provided by WebSphere Data Interchange. The default table can be overridden by specifying the name of an alternate alphanumeric data validation table in the Application Defaults profile. An alphanumeric data validation table specified in the Application Defaults profile will become the default table.</p> <p>For performance considerations, this field must have limited use. This is because it takes time to load an alphanumeric data validation table. When this field is used, the data validation table will replace the default alphanumeric data validation table in memory. After translation has been completed, there is a likelihood that the default alphanumeric data validation table, or another alphanumeric data validation table, will need to be loaded.</p> <p>This field is only used when the validation level is set to 2. Essentially, this table is used to define the set of valid characters for the alphanumeric data type (data type "AN").</p> |

## Send and receive usages

Table 33. Send map usages WDI options tab field descriptions (continued)

| Fields                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Character Data Validation Table | <p>This field is used to specify the name of an alternate character data validation table that is to be used for validating character data during translation. If a character data validation table is specified in this field, it will be used for the translation instead of the default. When this field is blank, the default character data validation table will be used.</p> <p>The character data validation table is defined as a Code List in the EDI Standards Functional Area. CHARSET is the name of the default table provided by WebSphere Data Interchange. The default table can be overridden by specifying the name of an alternate character data validation table in the Application Defaults profile. A character data validation table specified in the Application Defaults profile will become the default table.</p> <p>For performance considerations, this field must have limited use. This is because it takes time to load a character data validation table. When this field is used, the data validation table will replace the default character data validation table in memory. After translation has been completed, there is a likelihood that the default character data validation table, or another character data validation table, will need to be loaded.</p> <p>This field is only used when the validation level is set to 2. Essentially, this table is used to define the set of characters that can appear in an EDI Standard Transaction, excluding delimiters.</p> |
| Enforce Record Hierarchy        | <p>Use this check box to indicate whether you want WebSphere Data Interchange to issue an error if application data is passed to WebSphere Data Interchange out of the hierarchical sequence defined by the Data Format. A marked check box indicates that an error is issued when application data is passed to WebSphere Data Interchange out of the hierarchical sequence defined by the Data Format. An unmarked check box indicates an error is not issued under this condition.</p> <p>When marked, WebSphere Data Interchange automatically creates as many parent components as necessary to satisfy the hierarchical definition. The created components will be initialized with blanks.</p> <p>WebSphere Data Interchange will not issue an error for components that only exist for grouping other components. If a component does not contain any Fields, then its absence will not result in an error.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Record Must Produce Data        | <p>Use this check box to indicate whether WebSphere Data Interchange issues an error if either of the following occurs:</p> <ul style="list-style-type: none"> <li>• A Data Format Record associated with a loop or repeating Segment was provided, but it did not generate any data as output. For example, if all the Data Format Fields in the Record mapped to the EDI Standard Segment contained blank values.</li> <li>• A Data Format Record is provided and that Record is the sole source of data for an EDI Standard Segment but no standard data was generated.</li> </ul> <p>A marked check box indicates that an error is issued. An unmarked check box indicates that an error is not issued.</p> <p><b>Note:</b> This field is not used if you are using special hierarchical loop support.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

Table 33. Send map usages WDI options tab field descriptions (continued)

| Fields                            | Description                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Control Numbers by Transaction ID | When the Control Numbers by Transaction ID field is marked, WebSphere Data Interchange will assign control numbers based on the values for the sender/receiver pairing and specific EDI Standard Transaction combination being processed. If the box is not marked, control numbers are assigned based on the sender/receiver pairing alone. |

14. Click **Save** on the toolbar. WebSphere Data Interchange saves the new Send Map Usage to the System.
15. Close the editor when you are done.

## Creating a receive usage

In most cases, you do not need to create a Receive Map. Instead, you create a Data Transformation Map. Data Transformation Maps have effectively replaced Receive Maps. Receive Maps remain supported for compatibility with earlier version of WebSphere Data Interchange only. You can still create Receive Maps if you need to. However, there might come a point in time when Receive Maps must be converted to Data Transformation Maps. Receive Maps transform an EDI Standard Transaction into a proprietary Data Format document. Often, an existing map can be used with conditional mapping commands to produce similar documents with minor variations, perhaps based on the trading partner the source document is being received from. If there is no similar existing map, then a new map must be created. Figure 13 shows the Send Map Usage editor.

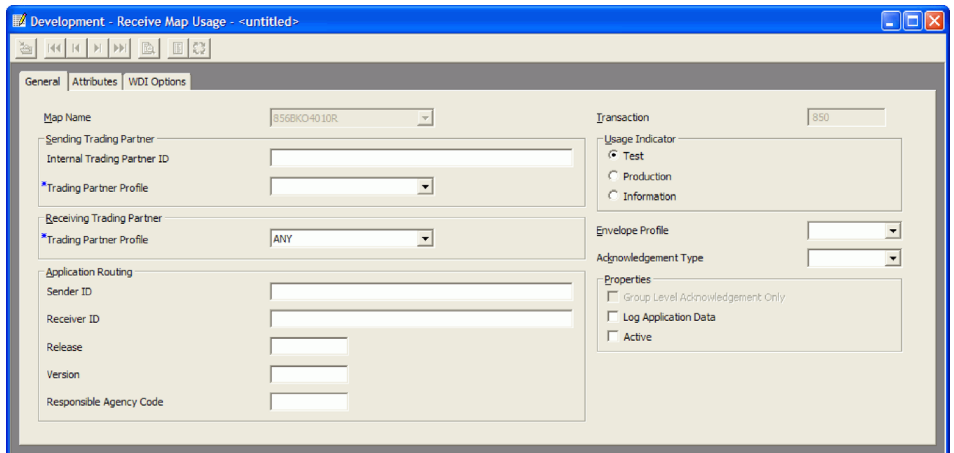


Figure 13. Receive Map Usage editor

To create a Receive map usage:

1. Open the Receive Map Usages list window. This can be accomplished by any of the following means:
  - Select the Open Functional Area action on the File menu. This opens the Open Functional Area dialog. Select Rules and Usages and then click OK.

## Send and receive usages

The Rules and Usages Functional Area window will open containing list windows for Map Rules, Send Map Usages, and Receive Map Usages. Click on the Receive Map Usages tab to display the Receive Map Usages list window. The list window contains a list of all Receive Map Usages selected by the current Query.

Alternately you can click Rules and Usages on the navigator bar to open the Rules and Usages Functional Area window.

- Select the Open Query List action on the File menu. This will open the Query list window dialog. Select the Rules and Usages Functional Area, and then select Receive Map Usages. Finally, select the Query you wish to execute and Run.

A window opens containing the Receive Map Usages list window. The list window contains a list of all Receive Map Usages selected by the Query.

- Open the Receive Maps list window by clicking Mapping on the WebSphere Data Interchange Client navigation bar. Click on the Receive Maps tab to display the Receive Maps list window. Select the Receive Map associated with the Receive Map Usage you wish to create. Click Rules and Usages on the toolbar or select Rules and Usages from the Actions menu.

A window opens containing the Receive Map Usages list window. It contains a list of all Receive Map Usages associated with the selected Send Map.

- Open the Trading Partners list window by clicking Trading Partner on the WebSphere Data Interchange Client navigation bar. Click on the Trading Partners tab to display the Trading Partners list window. Select the Trading Partner profile associated with the Receive Map Usage you wish to create. Click Rules and Usages on the toolbar or select Rules and Usages from the Actions menu.

A window opens containing the Receive Map Usages list window. It contains a list of Receive Map Usages associated with the selected Trading Partner profile.

2. Click New on the list window toolbar.

The Receive Map Usage editor opens with the **General** tab in front.

3. On the **General** tab, select the name of the map that to associated with this Receive Map Usage.

**Note:** When you select a map, then the Transaction field is filled in automatically using the map's source document. This cannot be changed. When the Receive Map Usages list window is opened from the Receive Maps list window or the Receive Map editor, the Map Name and Transaction fields are filled in and cannot be changed .

4. Enter an internal trading partner ID. This field is optional.
5. Select the name of the sending Trading Partner profile.

Set this field to an ampersand (&) when a generic Receive Map Usage is being defined.

**Note:** This field is pre-filled with the name of the selected Trading Partner profile when the selected Trading Partner profile is trading partner type EDI trading partner. You cannot change the value in this case. When the Receive Map

Usages list window is opened from the Trading Partners list window or the Trading Partner editor, this field might be pre-filled with the name of the selected Trading Partner profile. This occurs when the selected Trading Partner profile is trading partner type EDI Trading Partner. You will not be able to change this value in this case.

6. Select the name of the receiving Trading Partner profile.

**Note:** When the Receive Map Usages list window is opened from the Trading Partners list window or the Trading Partner editor, this field is pre-filled with the name of the selected Trading Partner profile. This occurs when the selected Trading Partner profile is trading partner type "Application Trading Partner". You cannot be able to change this value in this case.

7. In the Usage Indicator pane of the tab, indicate whether the Receive Map Usage is used when the source Transaction is a production, test, or information document. This field is used to help determine which Receive Map Usage is used to identify the Receive Map to use to translate the source Transaction. Documents being translated are classified as production, test or information documents. For a Transaction, the classification is determined from a flag in the EDI Standard Envelope associated with the Transaction.

**Note:** A production Receive Map Usage can be used for test Transactions or information Transactions when the Production Usage - Test Message check box in the Application Defaults profile is marked.

8. Indicate whether you want WebSphere Data Interchange to save an image of the application data in the Event Log.  
A marked Log Application Data check box indicates that generated application data is logged to the Event Log. An unmarked check box indicates that generated application data is not logged to the Event Log.
9. Indicate if the Receive Map Usage is active. Inactive Receive Map Usages are not used when trying to determine what Receive Map is used to translate a document.  
A Receive Map Usage can be active or inactive. Inactive Receive Map Usages are used when determining which Receive Map to use in the translation of a document. For each document, sending Trading Partner profile, receiving Trading Partner profile, application sender and receiver ID, release, version, and responsible agency code combination, there can only be one active production Receive Map Usage, one active test Receive Map Usage, and one active information Receive Map Usage. During translation, a search is performed to locate an active Receive Map Usage with a classification corresponding to the source Transaction. An error is issued during translation if an active Receive Map Usage for the source Transaction cannot be located.
10. In the Application Routing pane of the tab, enter information into the fields if these fields are to be used to help determine which Receive Map Usage is to be used to translate the source Transaction. See Table 34 on page 228 for a description of the fields.

## Send and receive usages

Table 34. Application Routing field descriptions

| Field     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Sender ID | <p>The application sender ID identifies the specific sender of the Transaction within the trading partner, such as a department number. If you enter a value in this field, do not enter a value in the application receiver ID field.</p> <p>When a Transaction is going to be translated, WebSphere Data Interchange looks for a Receive Map Usage with a matching application sender ID to help determine the correct Receive Map Usage. If there is no matching Receive Map Usage found, WebSphere Data Interchange looks for a Receive Map Usage with a matching application receiver ID to help determine the correct Receive Map Usage. If no matching Receive Map Usage is found, WebSphere Data Interchange uses a blank application sender and receiver ID to determine the correct Receive Map Usage. Therefore, you can define several active Receive Map Usages based on who the application sender ID or application receiver ID.</p> <ul style="list-style-type: none"><li>• If functional groups are present in the interchange being received, the application sender ID is taken from the Data Element with the data type AS.</li><li>• If functional groups are not present, the application sender ID is taken from the Data Element with data type RS.</li><li>• If the Data Element with data type RS is defined but contains blanks or contains no data, the application sender ID is taken from the Data Element with data type IS.</li><li>• If the Data Element with data type RS is not defined, the application sender ID is taken from the Data Element with data type AS.</li><li>• If the Data Element with data type AS is not defined, the application sender ID is taken from the Data Element with data type IS.</li></ul> <p>For existing Receive Map Usages, this field cannot be changed. If you need to change this field on an existing Receive Map Usage, you must copy the Receive Map Usage and then delete the original Receive Map Usage. This field can be changed when copying a Receive Map Usage.</p> |

Table 34. Application Routing field descriptions (continued)

| Field         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Receiver ID   | <p>The receiver ID identifies the specific receiver of the translated document within your company, such as a department number. If you enter a value in this field, do not enter a value in the application sender ID field.</p> <p>When a Transaction is going to be translated, WebSphere Data Interchange looks for a Receive Map Usage with a matching application sender ID to help determine the correct Receive Map Usage. If there is no matching Receive Map Usage found, WebSphere Data Interchange looks for a Receive Map Usage with a matching application receiver ID to help determine the correct Receive Map Usage. If no matching Receive Map Usage is found, WebSphere Data Interchange uses a blank application sender and receiver ID to determine the correct Receive Map Usage. Therefore, you can define several active Receive Map Usages based on who the application sender ID or application receiver ID.</p> <ul style="list-style-type: none"> <li>• If functional groups are present in the interchange being received, the application receiver ID is taken from the Data Element with the data type AR.</li> <li>• If functional groups are not present, the application receiver ID is taken from the Data Element with data type RR.</li> <li>• If the Data Element with data type RR is defined but contains blanks or contains no data, the application receiver ID is taken from the Data Element with data type IR.</li> <li>• If the Data Element with data type RR is not defined, the application receiver ID is taken from the Data Element with data type AR.</li> <li>• If the Data Element with data type AR is not defined, the application receiver ID is taken from the Data Element with data type IR.</li> </ul> <p>For existing Receive Map Usages, this field cannot be changed. If you need to change this field on an existing Receive Map Usage, you must copy the Receive Map Usage and then delete the original Receive Map Usage. This field can be changed when copying a Receive Map Usage.</p> |
| Release field | <p>Use this field in addition to the application sender ID or application receiver ID fields to provide different Receive Map Usage selection capabilities. It is expected, but not necessary, that this field is combined with the agency and version fields so that a different Receive Map Usage can be selected based on the defining agency, release and version of the Transaction being translated.</p> <p>For example, it is possible to have a separate map for a TDCC version 3 810 and an X12 version 3 810 from the same trading partner. The data entered into this field must exactly match the release when taken from the Transaction being translated other than trailing blanks, which are not significant.</p> <p>For existing Receive Map Usages, this field cannot be changed. If you need to change this field on an existing Receive Map Usage, you must copy the Receive Map Usage and then delete the original Receive Map Usage. This field can be changed when copying a Receive Map Usage.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

## Send and receive usages

Table 34. Application Routing field descriptions (continued)

| Field                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Version                 | <p>Use this field in addition to the application sender ID or application receiver ID fields to provide different Receive Map Usage selection capabilities. It is expected, but not necessary, that this field will be combined with the release and agency fields so that a different Receive Map Usage can be selected based on the defining agency, release and version of the Transaction being translated.</p> <p>For example, it is possible to have a separate map for a TDCC version 3 810 and an X12 version 3 810 from the same trading partner. The data entered into this field must exactly match the version when taken from the Transaction being translated other than trailing blanks, which are not significant.</p> <p>For existing Receive Map Usages, this field cannot be changed. If you need to change this field on an existing Receive Map Usage, you must copy the Receive Map Usage and then delete the original Receive Map Usage. This field can be changed when copying a Receive Map Usage.</p> |
| Responsible Agency Code | <p>Use this field in addition to the application sender ID or application receiver ID fields to provide different Receive Map Usage selection capabilities. It is expected, but not necessary, that this field will be combined with the release and version fields so that a different Receive Map Usage can be selected based on the defining agency, release and version of the Transaction being translated.</p> <p>For example, it is possible to have a separate map for a TDCC version 3 810 and an X12 version 3 810 from the same trading partner. The data entered into this field must exactly match the agency when taken from the Transaction being translated other than trailing blanks, which are not significant. For existing Receive Map Usages, this field cannot be changed. If you need to change this field on an existing Receive Map Usage, you must copy the Receive Map Usage and then delete the original Receive Map Usage. This field can be changed when copying a Receive Map Usage.</p>        |

- Identify the Envelope Profile if a functional acknowledgement is required.
 

The values that appear in this list are based upon the EDI Standard Transaction that is being translated. When the Transaction is UN/EDIFACT (UNB/UNZ), this field will list all E Envelope profiles. I Envelope profiles are listed when Transaction is Interchange Control Segments (ICS/ICE). When the Transaction is Trade Data Interchange (STX/END), this field will list all T Envelope profiles. When the Transaction is Uniform Communication Standard (BG/EG), this field will list all U Envelope profiles. All X Envelope profiles will be listed when the Transaction is X12 (ISA/IEA).

Select the Envelope profile to use when creating an Envelope related to the functional acknowledgement generated for the received EDI Standard Transaction. If the field is left blank, the Envelope profile with the same name as the EDI Standard Dictionary associated with the source EDI Standard Transaction will be used.



A generic Envelope profile name can be specified in this field. You can specify a generic Envelope profile name by entering an ampersand (&) followed by a 1 to 6 character base name. During translation, the Envelope profile suffix from the Trading Partner profile will be appended to the base name to dynamically determine the name of the Envelope profile to use when enveloping the document.

12. Identify which functional acknowledgement to generate in the Acknowledgement Type field if a functional acknowledgement is required.
13. Indicate if an outbound EDI functional acknowledgement is generated for inbound EDI data. A group level only functional acknowledgement is generated if no errors exist in the inbound EDI data.

**Note:** This field is only available after a an acknowledgement type is selected in the Acknowledgement Type field.

A marked Group Level Acknowledgement only check box indicates that an outbound EDI functional acknowledgement is generated. An unmarked check box indicates that no acknowledgement is generated.

14. Click on the Attributes tab and complete any required fields there. See Table 35 on page 232 for a description of the fields.

Table 35. Receive map usages Attributes tab field descriptions

| Field                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Document Destination Type | <p>This field along with the Document Destination Name field identifies the destination of the output document. For all platforms, the output document can be written to a file or to a WebSphere MQ message queue. In addition, the CICS platform can invoke an application to process the output document.</p> <p>A value here overrides the Document Destination Type field in the Data Format.</p> <p>The Receive Map used to translate the Transaction can specify what happens to the resulting document. This occurs when a value is specified in the DIAPPTYPE special variable. If a value occurs in the variable, it overrides the value specified here and in a Data Format.</p> <p>You can select a value to indicate the type of destination that is used for documents produced using this Receive Map Usage if you wish to specify a default output destination.</p> <p>All values apply to the WebSphere Data Interchange Server on the CICS platform. For all other server platforms, WebSphere MQ Message Queue is the only applicable value. If this field is left blank in CICS, the default is Temporary Storage Queue (Auxiliary Storage). A blank value on a z/OS system indicates the output document is written to a file and the value in the Document Destination Name field identifies a ddname. For all other platforms, a blank value indicates the output document is written to a file and the value in the Document Destination Name field identifies a logical file name. Specifying a value that is not applicable for the platform the WebSphere Data Interchange Server is running on results in the output document being written to a file.</p> <p>If the value in this field indicates a WebSphere MQ message queue is used, then the Document Destination Name field identifies the name of a MQSeries Queue profile. The MQSeries Queue profile identifies the WebSphere MQ message queue used to write the document during translation.</p> <p>On the CICS platform, the document can be routed to a response program or response transaction. These are CICS applications designed to accept and process the output document produced by WebSphere Data Interchange. Select CICS Program Name to indicate the Document Destination Name field identifies the name of a CICS program. Select CICS Transaction ID to indicate the Document Destination Name field identifies the name of a CICS transaction. See the <i>WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide</i>, SC34-6217-01 for more information about response programs and response transactions.</p> |

Table 35. Receive map usages Attributes tab field descriptions (continued)

| Field                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Document Destination Type | <p>This field along with the Document Destination Name field identifies the destination of the output document. For all platforms, the output document can be written to a file or to a WebSphere MQ message queue. In addition, the CICS platform can invoke an application to process the output document.</p> <p>A value here overrides the Document Destination Type field in the Data Format.</p> <p>The Receive Map used to translate the Transaction can specify what happens to the resulting document. This occurs when a value is specified in the DIAPPTYPE special variable. If a value occurs in the variable, it overrides the value specified here and in a Data Format.</p> <p>You can select a value to indicate the type of destination that is used for documents produced using this Receive Map Usage if you wish to specify a default output destination.</p> <p>All values apply to the WebSphere Data Interchange Server on the CICS platform. For all other server platforms, WebSphere MQ Message Queue is the only applicable value. If this field is left blank in CICS, the default is Temporary Storage Queue (Auxiliary Storage). A blank value on a z/OS system indicates the output document is written to a file and the value in the Document Destination Name field identifies a ddname. For all other platforms, a blank value indicates the output document is written to a file and the value in the Document Destination Name field identifies a logical file name. Specifying a value that is not applicable for the platform the WebSphere Data Interchange Server running on results in the output document being written to a file.</p> <p>If the value in this field indicates a WebSphere MQ message queue is used, then the Document Destination Name field identifies the name of a MQSeries Queue profile. The MQSeries Queue profile identifies the WebSphere MQ message queue used to write the document during translation.</p> <p>On the CICS platform, the document can be routed to a response program or response transaction. These are CICS applications designed to accept and process the output document produced by WebSphere Data Interchange. Select CICS Program Name to indicate the Document Destination Name field identifies the name of a CICS program. Select CICS Transaction ID to indicate the Document Destination Name field identifies the name of a CICS transaction. See the <i>WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide</i>, SC34-6217-01 for more information about response programs and response transactions.</p> |
| Application Password      | This field is not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

- Click on the **WDI Options** tab. Identify any processing options that are used when processing a Transaction using this Receive Map Usage.

Table 36. Receive map usages WDI Options tab field descriptions

| Field                                                                                                                                                                                                                                                                                                                                                                                                                                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----------------------------------|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|---------------------------------------------|--|------------------------------------------------------------------------------------------------------------------------------|--|-------------------------------------------------------------------------|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Pre-Translation Exit Routine                                                                                                                                                                                                                                                                                                                                                                                                          | <p>This field can be used to indicate that WebSphere Data Interchange is supposed to call a user-written exit routine before translating the document.</p> <p>If a pre-translation exit routine is called, select the name of a User Exit profile from the list. The User Exit profile identifies the exit routine that is called. Only User Exit profiles identified as pre-translation exit routines are listed in the list.</p> <p>See the <i>WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide</i>, SC34-6217-01 for additional information about exit routines.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| Validation Level                                                                                                                                                                                                                                                                                                                                                                                                                      | <p>Use this field to indicate the level of automatic validation you would like performed during the translation. The valid values are:</p> <table border="0"> <thead> <tr> <th data-bbox="505 631 565 656">Value</th> <th data-bbox="602 631 690 656">Meaning</th> </tr> </thead> <tbody> <tr> <td colspan="2" data-bbox="505 673 780 697"><b>Only Mandatory Validation</b></td> </tr> <tr> <td colspan="2" data-bbox="602 701 1220 835">No additional validation over the mandatory validation done by WebSphere Data Interchange to convert data between the EDI Standard data type and the target data type. With this level of validation, any Code Lists specified in the Receive Map or in an EDI Standard Data Element are ignored.</td> </tr> <tr> <td colspan="2" data-bbox="505 852 895 876"><b>Mandatory Validation Plus Code Lists</b></td> </tr> <tr> <td colspan="2" data-bbox="602 880 1186 956">Includes mandatory validation plus the use of the Code Lists specified in the Receive Map and in EDI Standard Data Elements.</td> </tr> <tr> <td colspan="2" data-bbox="505 973 1180 998"><b>Mandatory Validation Plus Code Lists Plus Data Type Verification</b></td> </tr> <tr> <td colspan="2" data-bbox="602 1001 1213 1187">Includes mandatory validation plus the use of the Code Lists specified in the Receive Map and in EDI Standard Data Elements. In addition, verification that the data values supplied are consistent with the Data Element's data type will occur. For instance, use this level of validation to verify that a Data Element that is supposed to contain a date has a valid date, or a numeric Data Element contains numeric data only.</td> </tr> </tbody> </table> | Value | Meaning | <b>Only Mandatory Validation</b> |  | No additional validation over the mandatory validation done by WebSphere Data Interchange to convert data between the EDI Standard data type and the target data type. With this level of validation, any Code Lists specified in the Receive Map or in an EDI Standard Data Element are ignored. |  | <b>Mandatory Validation Plus Code Lists</b> |  | Includes mandatory validation plus the use of the Code Lists specified in the Receive Map and in EDI Standard Data Elements. |  | <b>Mandatory Validation Plus Code Lists Plus Data Type Verification</b> |  | Includes mandatory validation plus the use of the Code Lists specified in the Receive Map and in EDI Standard Data Elements. In addition, verification that the data values supplied are consistent with the Data Element's data type will occur. For instance, use this level of validation to verify that a Data Element that is supposed to contain a date has a valid date, or a numeric Data Element contains numeric data only. |  |
| Value                                                                                                                                                                                                                                                                                                                                                                                                                                 | Meaning                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| <b>Only Mandatory Validation</b>                                                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| No additional validation over the mandatory validation done by WebSphere Data Interchange to convert data between the EDI Standard data type and the target data type. With this level of validation, any Code Lists specified in the Receive Map or in an EDI Standard Data Element are ignored.                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| <b>Mandatory Validation Plus Code Lists</b>                                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| Includes mandatory validation plus the use of the Code Lists specified in the Receive Map and in EDI Standard Data Elements.                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| <b>Mandatory Validation Plus Code Lists Plus Data Type Verification</b>                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| Includes mandatory validation plus the use of the Code Lists specified in the Receive Map and in EDI Standard Data Elements. In addition, verification that the data values supplied are consistent with the Data Element's data type will occur. For instance, use this level of validation to verify that a Data Element that is supposed to contain a date has a valid date, or a numeric Data Element contains numeric data only. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| Acceptable Error Level                                                                                                                                                                                                                                                                                                                                                                                                                | <p>Use this field to indicate what level of error will be acceptable for a translation to be considered successful. Valid values are:</p> <ul style="list-style-type: none"> <li>• Allow only translations with no errors.</li> <li>• Allow a translation that has no error or has only Data Element errors.</li> <li>• Allow a translation that has no error, has Segment errors, or has Data Element errors.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |       |         |                                  |  |                                                                                                                                                                                                                                                                                                   |  |                                             |  |                                                                                                                              |  |                                                                         |  |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |

Table 36. Receive map usages WDI Options tab field descriptions (continued)

| Field                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alphanumeric Data Validation Table | <p>This field is used to specify the name of an alternate alphanumeric data validation table that is to be used for validating alphanumeric character data during translation. The table identified in this field contains a subset of the characters found in the character data validation table. If an alphanumeric data validation table is specified in this field, it is used for the translation instead of the default alphanumeric data validation table. When this field is blank, the default alphanumeric data validation table is used.</p> <p>The alphanumeric data validation table is defined as a Code List in the EDI Standards Functional Area. ALPHANUM is the name of the default table provided by WebSphere Data Interchange. The default table can be overridden by specifying the name of an alternate alphanumeric data validation table in the Application Defaults profile. An alphanumeric data validation table specified in the Application Defaults profile becomes the default table.</p> <p>For performance considerations, limit the use of this field. This is because it takes time to load an alphanumeric data validation table. When this field is used, the data validation table will replace the default alphanumeric data validation table in memory. After translation has been completed, there is a likelihood that the default alphanumeric data validation table, or another alphanumeric data validation table, is loaded.</p> <p>This field is only used when the validation level is set to 2. Essentially, this table is used to define the set of valid characters for the alphanumeric data type (data type "AN").</p> |
| Character Data Validation Table    | <p>This field is used to specify the name of an alternate character data validation table that is to be used for validating character data during translation. If a character data validation table is specified in this field, it will be used for the translation instead of the default. When this field is blank, the default character data validation table will be used.</p> <p>The character data validation table is defined as a Code List in the EDI Standards Functional Area. CHARSET is the name of the default table provided by WebSphere Data Interchange. The default table can be overridden by specifying the name of an alternate character data validation table in the Application Defaults profile. A character data validation table specified in the Application Defaults profile will become the default table.</p> <p>For performance considerations, limit the use of this field. This is because it takes time to load a character data validation table. When this field is used, the data validation table will replace the default character data validation table in memory. After translation has been completed, there is a likelihood that the default character data validation table, or another character data validation table, will need to be loaded.</p> <p>This field is only used when the validation level is set to 2. Essentially, this table is used to define the set of characters that can appear in an EDI Standard Transaction, excluding delimiters.</p>                                                                                                                                                              |

## Send and receive usages

Table 36. Receive map usages WDI Options tab field descriptions (continued)

| Field                                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data Overlay Check                   | <p>Use this check box to indicate whether WebSphere Data Interchange issues an error when a data overlay condition occurs. A marked check box indicates that an error is issued when a data overlay condition occurs. An unmarked check box indicates that an error is not issued when a data overlay condition occurs. Data overlay conditions can occur if:</p> <ul style="list-style-type: none"> <li>• You are mapping data from a repeating Segment to a non-repeating application component.</li> <li>• More than one occurrence of a qualified Segment, loop, or Data Element is received when only one is expected.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Unexpected Data Element Check        | <p>This check box is used to indicate whether WebSphere Data Interchange issues an error when the source Transaction includes Data Elements that are not mapped. Mark this check box to indicate that an error occurs in this situation. Leave the check box unmarked when no error occurs.</p> <p>Unmapped Data Elements can indicate that your trading partner's application has changed and includes new Data Elements in mapped Segments.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Unexpected Segment Check             | <p>This check box is used to indicate whether WebSphere Data Interchange issues an error when the source Transaction includes Segments that are not mapped. Mark this check box to indicate that an error occurs in this situation. Leave the check box unmarked when no error occurs.</p> <p>Unmapped Segments can indicate that your trading partner's application has changed and includes new information.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Switch Application Routing IDs on FA | <p>When a Transaction is received in an EDI Standard Envelope, the Envelope contains an application sender ID and application receiver ID. When a functional acknowledgement is to be generated, WebSphere Data Interchange can use the received application sender ID and application receiver ID in the Envelope that is generated for the functional acknowledgement. Mark this check box to indicate the received sender ID and receiver ID are to be used in the envelope associated with the functional acknowledgement. Leave this check box unmarked if the sender ID and receiver ID used in the Envelope for the functional acknowledgement will be provided elsewhere (such as the Envelope profile specified for the functional acknowledgement).</p> <p>If this check box is marked:</p> <ul style="list-style-type: none"> <li>• The application sender ID in the received Envelope becomes the application receiver ID in the Envelope generated for the functional acknowledgement.</li> <li>• The application receiver ID in the received Envelope becomes the application sender ID in the Envelope generated for the functional acknowledgement.</li> </ul> |

Table 36. Receive map usages WDI Options tab field descriptions (continued)

| Field                             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inbound Envelope Used on FA       | <p>This check box is used to indicate whether the EDI Standard Envelope received with the Transaction being translated is used to generate the Envelope associated with a generated functional acknowledgement. Mark this check box to indicate that the Envelope received with the Transaction is to be used to generate the Envelope associated with the generated functional acknowledgement. Leave this check box unmarked if the Envelope associated with the functional acknowledgement is generated normally.</p> <p>If this check box is marked, the following tables indicate which elements received in the Envelope will be moved or switched to generate the Envelope for the functional acknowledgement:</p> <ul style="list-style-type: none"> <li>• UN/EDIFACT (UNB/UNZ) Envelope<br/>UNB01 to UNB01, UNB02 to UNB02, UNB03 to UNB06, UNB04 to UNB07, UNB05 to UNB08, UNB06 to UNB03, UNB07 to UNB04, UNB08 to UNB05, UNB15 to UNB15, UNB17 to UNB17, UNG02 to UNG04, UNG03 to UNG05, UNG04 to UNG02, UNG05 to UNG03, UNG09 to UNG09, UNG10 to UNG10, UNG11 to UNG11.</li> <li>• Interchange Control Segments (ICS/ICE) Envelope<br/>ICS02 to ICS02, ICS03 to ICS03, ICS04 to ICS06, ICS05 to ICS07, GS02 to GS03, GS07 to GS07, GS08 to GS08.</li> <li>• Trade Data Interchange (STX/END) Envelope<br/>STX01 to STX01, STX02 to STX02, STX03 to STX05, STX04 to STX06, STX05 to STX03, STX06 to STX04, STX12 to STX12.</li> <li>• Uniform Communication Standard (BG/EG) Envelope<br/>BG02 to BG03, BG03 to BG02, BG04 to BG05, BG05 to BG04, GS02 to GS03, GS03 to GS02, GS07 to GS07, GS08 to GS08.</li> <li>• X12 (ISA/IEA) Envelope<br/>ISA05 to ISA07, ISA06 to ISA08, ISA07 to ISA05, ISA08 to ISA06, ISA11 to ISA11, ISA12 to ISA12, GS02 to GS03, GS03 to GS02, GS07 to GS07, GS08 to GS08.</li> </ul> <p>The Envelope profile name specified on the Receive Map Usage will override the values obtained from the received Envelope.</p> |
| Control Numbers by Transaction ID | <p>When the Control Numbers by Transaction ID check box is marked, WebSphere Data Interchange will assign control numbers based on the values for the sender/receiver pairing and specific EDI Standard Transaction combination being processed. If the box is not marked, control numbers are assigned based on the sender/receiver pairing alone. Control numbers are generated for the functional acknowledgement that might be generated as a result of processing the source Transaction.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

16. Click Save on the toolbar. WebSphere Data Interchange saves the new Receive Map Usage to the system.
17. Close the editor when you are done.

## Send and receive usages

### Editing send and receive usages

Edit a send or receive usage when you need to change translation specifications.

1. In the Mapping list window, click the map for which you want to edit a send or receive usage.
2. Click View Usages on the tool bar.

Either the **Send Map Usages** tab or the **Receive Map Usages** opens. If you have created usages for this map, the existing usages display in the list window.

3. Double-click the send or receive usage you need to edit.  
The **General** opens.
4. Add, change, or delete entries as required.
5. When you have finished entering all values required in the send or receive usage, click Save on the tool bar to save the send or receive usage.

### Copying send or receive usages

The copy function duplicates a send or receive usage within the WebSphere Data Interchange system in which you are working. If you want to base a new send or receive usage on an existing one, for example, copy the existing trading send or receive under a new name and edit it to the new specifications. You can also copy a usage to a different map or to a new trading partner.

1. In the Mapping list window, select the map that is associated with the send or receive usage you wish to copy.
2. Click View Usages on the tool bar.

Either the **Send Map Usages** tab or the **Receive Map Usages** opens. If you have created usages for this map, the existing usages display in the list window.

3. Select the send or receive usage you wish to copy.
4. Select Copy from the Actions menu.

The Copy Send Usage or Copy Receive Usage window opens.

5. Select or enter new values into the field provided on the window.
6. Click OK.

WebSphere Data Interchange Client copies the send or receive usage using the information specified on the copy window.

WebSphere Data Interchange Client copies the send or receive usage using the information specified on the copy window. For information about deleting send or receive usages, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01.

### Defining generic send usages

Where multiple trading partners can use the same send usage definition and map, a generic send usage can be defined to WebSphere Data Interchange. When combined with a generic routing code supplied by the application, it provides the capability to define one or more generic usages, each of which can handle multiple trading partners.



The generic routing code is an optional, three-character code provided by the application to select the correct generic usage when no specific usage has been defined for the trading partner. The Generic Routing Code can be provided in one of three ways:

1. The Translator Control Block (TRCB) includes a three-character field for the generic routing code. The API calls can provide the routing code in the TRCB.
2. For C and D processing, the C record includes a field for the three-character generic routing code.
3. For raw data processing, an application field that contains the generic routing code can be specified when defining the data format. The application field is one to three characters. If less than three characters, the value is filled with blanks. If greater than three characters, it is truncated to three.

When defining send usages, an Internal Trading Partner ID that begins with an ampersand (&) indicates it is a generic usage. For a specific usage by routing code, specify an ampersand (&) followed by the three-character generic routing code that the application provides to select this usage.

If the application does not want to provide a routing code or does provide a routing code but wants to select a default generic usage, the Internal Trading Partner ID can be defined as a single (&) followed by blanks. This type of generic definition is the default definition and is selected when no routing code is provided, or when the routing code is provided but no specific usage is found, or when the routing code contains blanks.

The purpose of the generic routing code is to use a file containing one data format to generate more than one type of transaction (such as purchase orders and purchase-order changes). The generic routing code is required when the user wants to process transactions using different maps that reference the same data format.

For example, if the user wants to process purchase orders and purchase-order change documents that are defined using the same data format, the application can provide the following: a routing code of POR for a purchase order and WebSphere Data Interchange would select the usage with the Internal Trading Partner ID of &POR, which would reference a purchase order map; and a routing code of POC for a purchase-order change and WebSphere Data Interchange would select usage &POC, which would reference a purchase-order change map. An additional benefit when using this type of definition is that multiple documents can be included in the same raw data file if they all use the same data format.

### Defining generic receive usages

When transactions received from multiple trading partners can use the same mapping, a generic receive usage can be defined to WebSphere Data Interchange to handle multiple trading partners with a single usage and map.

When defining the generic receive usages, a trading partner nickname with only an ampersand (&) is a special form indicating that it is a generic usage. The generic usage is selected when the normal selection process using the trading partner nickname does not find a receive usage.

## Send and receive usages

Many generic receive usages can be defined as long as one of the listed match criteria is different (such as Application Sender, Application Receiver, Agency, Version, or Release, production/test).

---

## Creating fixed-to-fixed maps

Fixed-to-fixed translation is a method of translating application data from one format to application data of another format. This requires a Send map to direct the movement of data between data formats.

There are Mapping and Control String Export considerations. The Target Data Format definition used to create the Standard that is used in the fixed-to-fixed mapping process must be present in the Server Execution or Runtime data base. It is not automatically exported with the Map or Control String and must be exported in a separate export execution.

**Note:** It is recommended that a Data Transformation map be used to create a map from a data format to a data format instead of using a Send map. When using a Data Transformation map, you can specify the original data format definitions in the source and target documents, avoiding the need to convert a data format to an EDI standard.

To create a fixed-to-fixed map:

1. The target data format must be converted to an EDI standard. Begin by selecting the target data format from the Data Format list window.
2. Select Create Standard from Data Format from the Action menu to convert the target data format into an EDI standard.

**Note:** The name of the generated EDI standard is taken from the Application File/Queue name field on the **General** tab of the Data Format editor. This conversion only needs to be done once when the target data format is defined. If the target data format changes, the Create Standard from data format can be repeated to delete the old EDI standard and create a new one.

An EDI standard is created. You can view this using the EDI standards related list windows and editors.

3. Create a Send map using the source data format and the target EDI standard just created. See “Creating a send or Receive map” on page 201.
4. Create send usages as needed. See “Specifying send and receive usages” on page 215.
5. Compile the control string for the map. See “Compiling control strings” on page 116.

---

## Chapter 14. Advanced send and Receive mapping

This appendix describes how to map your application data to an EDI standard transaction set. This appendix assumes that you are familiar with your application data layout and have already defined your application data to WebSphere Data Interchange. It also assumes you are familiar with the EDI standard transaction set you are using.

---

### Using accumulators

Accumulators are special fields that keep running totals or accumulate numeric data. Accordingly, WebSphere Data Interchange's accumulators can add data to an EDI transaction that does not occur in an application database and vice versa. WebSphere Data Interchange Client supports global and transaction accumulators. The scope of a global accumulator is an entire translation session. The scope of a transaction accumulator is a single transaction.

You can use accumulators to count occurrences of an event, such as counting the detail line items in a purchase order to provide a hash total. You can also use accumulators to total fields for control purposes, such as totaling the quantity field to cross check the number of items sent or received.

Accumulators can apply to individual transactions or to all transactions in a translation session. To map both an accumulator and a received value for the same element, use the Repeat action to create another occurrence of the element mapping. Then map one occurrence from the data element to a field and the other occurrence from the accumulator to a field. Each element mapping can support up to four accumulators.

For send transactions, accumulator actions will not be processed unless one of the following occurs:

- Data is generated for the EDI standard data element.
- The variable is mapped.

For receive transactions, accumulator actions will not be processed unless one of the following occurs:

- The data element associated with the accumulator is received.
- The accumulator is mapped and at least the segment containing the data element is received.

To map both an accumulator and a received value for the same data element, map the data element to a field. Then click Repeat in the Mapping Data Element editor to create another mapping occurrence of the data element. In the new mapping occurrence of the data element, map from the accumulator to a field.

Accumulators have the following limitations:

- Each accumulator holds a maximum of 31 digits.
- Each data element mapping can support up to 4 accumulators.

## Using accumulators

- You can use up to 10 transaction accumulators for a transaction.
- You can use up to 10 global accumulators for an entire translation session.

## Accumulator types

WebSphere Data Interchange supports transaction accumulators and global accumulators, as follows.

*Table 37. WebSphere Data Interchange accumulator types*

| Name  | Type                    | Description                                                                                                                                                                                      |
|-------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T0-T9 | Transaction Accumulator | Applies only to one transaction and are reset at the beginning of each transaction. You can use a maximum of 10 per transaction. Each accumulator holds a maximum of 31 binary digits.           |
| G0-G9 | Global Accumulator      | Applies to entire translation session and are reset at the beginning of each translation session. You can use a maximum of 10 per session. Each accumulator holds a maximum of 31 binary digits. |

## Accumulator actions

WebSphere Data Interchange supports the following accumulator actions.

*Table 38. Accumulator actions*

| This action. . .                          | Does this:                                                                                                                                                                                                               |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Increment the accumulator                 | Adds 1 to the value stored in the accumulator.                                                                                                                                                                           |
| Map the accumulator                       | Maps the value stored in the accumulator to the data element for Send maps and to the application field for Receive maps.                                                                                                |
| Zero the accumulator                      | Sets the accumulator value to 0.                                                                                                                                                                                         |
| Map the accumulator and then increment it | Maps the value stored in the accumulator to the data element for Send maps and to the application field for Receive maps, and then adds 1 to that value.<br><br>Only one accumulator can be mapped on any given mapping. |
| Increment the accumulator and then map it | Adds 1 to the value stored in the accumulator, and then maps the value stored in the accumulator to the data element for Send maps and to the application field for Receive maps.                                        |
| Add to the accumulator and then map it    | Adds the value of an element or field, and then maps the value stored in the accumulator to the data element for Send maps and to the application field for Receive maps.                                                |
| Map the accumulator and then add to it    | Maps the value stored in the accumulator to the standard data element for Send maps and to the application field for Receive maps, and then adds the value of a data element or field to it.                             |

## Adding an accumulator to a map

You set up an accumulator through the Mapping Data Element editor, as follows.

1. Double-click an element mapping in an EDI transaction that has been mapped to a field from a data format.
2. Double-click a data element in an EDI transaction that has been mapped to a field from a data format or on a field that has been mapped onto a data element.  
The Mapping Data Element editor opens.
3. Select an accumulator from the Accumulators list.  
Accumulator values are described in Table 37 on page 242.
4. Select an action for the accumulator from the Actions list.  
Only actions that are valid for this data element or field display in the list. Actions are described in Table 38 on page 242.
5. Complete any other mapping you need from the Mapping Data Element editor and click OK.

For outgoing data, accumulator actions are not processed unless:

- Data is generated for the data element or segment, or
- The accumulator is mapped.

For incoming data, the accumulator actions are not processed unless:

- The data element associated with the accumulator is received, or
- The accumulator is mapped and at least the segment containing the data element is received.

---

## Using literals

For send transactions, literals let you supply data that is not in your application data. For receive transactions, literals let you supply data required by your application that is not received with the standard data. WebSphere Data Interchange offers a variety of options for using literals. This section provides details on each option available, including rules for use, keywords, and syntax.

Transactions are always processed starting with the first data element of the first segment and proceeding to the last data element of the last segment, as documented in the standard. This is true for both send and receive processing.

### Using literals for Send mapping

When mapping literals for sending data, you can map both an application field and a literal to the same data element. The literal value is used if one of the following conditions occur:

- The record containing the application data field was provided, but the application data field does not supply a value or contains all blanks.
- The application data field supplies a value that does not match a value in the validation or translation table specified in the mapping.
- The application data field supplies a value, but the conversion from the application data type to the EDI standard data type fails.

## Using literals

WebSphere Data Interchange attempts a conversion for any application data field defined as a numeric field. The conversion removes leading and trailing blanks, leading zeros before the decimal, trailing zeros after the decimal, and changes the decimal point if the application decimal notation is different from the EDI standard decimal notation. If the data types are numeric, the conversion fails if the data contained anything other than a number or a decimal point. The literal value is used if the conversion fails. See “Validation during mapping” on page 282 for more information.

- If a translation table is specified, the literal value is checked against this table. If no matching entry is found, the translator logs a warning message in the event log, then uses the literal value. For information about the event log, see Table 15 on page 98.

## Segment creation for Send mapping

When an EDI standard segment is mapped with a combination of literal values and application data, the application data determines if the segment is produced. When the values from the application data produce data for the segment, the EDI standard segment is created. When the values in the application data do not produce data for the segment, such as when the records are not found, the application fields contain all blanks or zeros, or the application field values are not found in the associated validation or translation tables, then the EDI standard segment is not created. An exception is when zeros are passed to a mandatory numeric data element, in which case the segment is created. If this is not desirable, then pass blanks in the application data instead, or use the logic information presented later in this section to suppress the segment.

In some cases it might be desirable to suppress a segment altogether depending on input application data values. It is possible to suppress a segment using **IF** logic. When all contributing data element mappings for a segment contain conditional **IF** statements and all conditions are false, the segment is not produced. A contributing data element mapping is one that can produce output in the corresponding data element; such as specifying an application field name or a literal value, or specifying **USE** on a variable. Noncontributing data element mappings, such as **SAVE** or **SET**, do not have any effect on segment creation. See Table 39 on page 246 for details on **IF**, **USE**, **SAVE**, and **SET** usage.

**Note:** If the segment being suppressed begins a loop of segments, the entire loop is suppressed.

## Using literals for Receive mapping

If a data element mapping provides a literal, the literal value is used if one of the following conditions occur:

- The EDI standard data includes the segment being mapped, but the data element within the segment does not supply a value.
- The EDI standard data element supplies a value that does not match a value in the validation or translation table specified in the mapping.
- The EDI standard data element supplies a value, but the conversion from the EDI standard data type to the application data type fails. WebSphere Data Interchange attempts a conversion for any EDI standard data field defined as a numeric field. If

the data types are numeric and the EDI standard data field contains nonnumeric characters, the conversion would fail and the literal value would be used.

- The EDI standard data element supplies a value, but the &FORCE special literal is used to force the literal into the application field regardless of the EDI standard data element's contents.

## Format of literal data

When using literals in send or Receive mapping:

- For data types *BN*, *Bn*, *HX*, *Hn*, *IT*, *In*, *Ln*, *PD*, *Pn*, *ZD*, and *Zn*, the translator converts the literal before placing it in the standard data (send) or the application data (receive). For sending, it converts the literal to character data. For receiving, it converts the literal to the application data type. For example, if the data type is binary, the translator converts the literal to binary, then moves it to the application field.
- Do not type a decimal point when it is implied. For example, if you want to use a default value of 9.99 for a field defined as data type P2 (packed number with two implied decimal positions), enter 999 as the value of the literal.
- Enter literal values for hexadecimal fields as hexadecimal strings. For example, if the application field is defined as a one-byte hexadecimal field and you want to use a default value of X'FF', enter FF as the value of the literal. For receiving, the translator converts each two bytes of literal value to a single byte of application data.
- You can specify a literal value of zero to move a value into the standard field. WebSphere Data Interchange generally removes leading zeros from an application field so that an application field containing nothing but blanks or zeros will not result in a value for the data element. A value of zero is treated the same as all other literal values when determining if a segment is created. The &ZEROSIG special literal can also be used to indicate that zeros within the application field are significant.

In translation and validation tables, enter numeric values left-justified and formatted according to the application data format. For example, if the data format defines a field as R2, enter the value 7 as 7.00 or the value 7.1 as 7.10.

## Accumulator literals

Accumulator literals access the local and global accumulators. You can map an accumulator using the accumulator action, or you can map it using the literal associated with the accumulator. By using literals, you can exit to gain control. An accumulator has a data type of R. The literals are:

**&Tn** (Where *n* can be 0 through 9) identifies the accumulators T0 through T9.

**&Gn** (Where *n* can be 0 through 9) identifies the accumulators G0 through G9.

## Conditional processing of literals

Conditional processing lets you define how you want data processed, based on rules you establish. The following terms are used in the conditional processing:

### Named variable

A name used to represent data whose value can be changed while a program is running. You supply the name you want to use.

## Using literals

### Expression

A sequence of instructions which can consist of named variables, literals, operators, and constants. When processed, this sequence of instructions provides a single value.

### Constant

A value that does not change.

### Operator

A symbol that represents an operation to be done. WebSphere Data Interchange uses arithmetic operators, Boolean operators, comparison operators, unary operators, relational operators, and special operators.

### Operation

An action performed on one or more data items such as multiplying, comparing or moving.

**Value** A data value you provide. You can use any of the special literals that provide a data value; for example, &DATE, &TIME, &E, &ICN. This includes &T0 through &T9 and &G0 through &G9 to get the values for global and local accumulators.

### Default value

A default data value you provide. This value is not enclosed in quotation marks (""). You can use any of the special literals that provide a data value; for example, &DATE, &TIME, &E, &ICN. This includes &T0 through &T9 and &G0 through &G9 to get the values for global and local accumulators.

**Note:** When using conditional processing on a data element used to qualify a loop or repeating segment, the first occurrence of the qualifying data element mapping must specify a literal or application data field.

## Literal keywords

Table 39. Literal keywords

| Keyword  | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &ACFIELD | S/R | <p><b>Syntax:</b> &amp;ACFIELD</p> <p>Substitutes the application control value established by the mappings of the AC field from the data format or the concatenation of the fields specified during creation of the transaction mapping.</p> <p><b>Note:</b> Literals specified with &amp;LIT and variables specified with &amp;VAR in the mapping are not available in the AC field until the end of the translation process and will not appear in the AC field data that is moved using the &amp;ACFIELD keyword.</p> |



Table 39. Literal keywords (continued)

| Keyword  | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &ASSERTn | S/R | <p><b>Syntax:</b> &amp;ASSERTn(<i>expression</i>)</p> <p>The action associated with this literal is only executed if the <i>expression</i> is false and the assertion level is not greater than <i>n</i>. The &amp;ASSERT keyword can be combined with:</p> <ul style="list-style-type: none"> <li>• A mapping between an application data field and EDI standard data elements</li> <li>• The &amp;SET, &amp;SAVE, and &amp;USE keywords</li> <li>• An error condition; for example, &amp;ASSERTn followed by &amp;ERR.</li> </ul> <p>The differences between &amp;ASSERT and &amp;IF are:</p> <ul style="list-style-type: none"> <li>• &amp;ASSERT is a statement about the transaction that is expected to be true. For example, the number of items processed in the transaction equals the total number of items claimed to be in the transaction (value from CTT segment). If the &amp;ASSERT is not true, a special action takes place. An &amp;ASSERT is usually associated with an &amp;ERR condition, but this is not required.</li> <li>• With &amp;IF, if the expression is true, a special action takes place. An &amp;IF is usually associated with a mapping or named variable, but this is not required.</li> <li>• &amp;ASSERT has 10 levels (&amp;ASSERT0 through &amp;ASSERT9). Level 9 assertions are always executed. Assertions at level 0 (&amp;ASSERT0) through 8 are controlled by the assertion level used to start the translation. Thus, it is possible to turn off assertions, but &amp;IF conditions are always checked.</li> </ul> <p>When using the API, the assertion level is set in the <b>ASSERTLVL</b> field of the translator control block. When using the WebSphere Data Interchange Utility, the assertion level is set with the ASSERTLVL keyword on the PERFORM command. See the <i>WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide</i>, SC34-6217-01 for additional information about the API.</p> <p>See “Expressions” on page 258 for more information.</p> <p>For an example of using this keyword, see “Example 8” on page 274.</p> |
| &DATE    | S/R | <p><b>Syntax:</b> &amp;DATE</p> <p>Substitutes the system date. The length of the date field in the EDI standard data (send) or application data (receive) determines whether the date is formatted as <i>yyyymmdd</i> or <i>yymmdd</i>. The date is then edited as requested by the date edit specified in mapping.</p> <p>You can use the &amp;DATE keyword as source data for any of the EDI standard data types.</p> <p>You can also combine the &amp;DATE keyword with the &amp;IFDATA, &amp;IFNODATA, or &amp;FORCE keywords.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## Using literals

Table 39. Literal keywords (continued)

| Keyword   | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &DEFERRED | S   | <p><b>Syntax:</b> &amp;DEFERRED &amp;USE variable</p> <p>Allows you to signal that at a later time a value will be set using the same name as specified in the &amp;DEFERRED &amp;USE mapping. When the value is set, the data is put into the mapped segment.</p> <p>For example, if you had an HDR segment that contained a total number of items field, you can use &amp;DEFERRED &amp;USE TOTITEMS while mapping the HDR segment field.</p> <p><b>Note:</b> It is not recommended that the value be set within the same segment mapping. This will produce unexpected results. If the entire segment is mapped based on the use of deferred mapping, the segment can be produced without data.</p> <p>At the end of the mapping, you would then repeat map some field with &amp;SET TOTITEMS X, which will move the literal value 'X' into the HDR segment field. If X is used as a variable, the mapping would be &amp;SET TOTITEMS &amp;E(X), which would move the contents of the variable X into the HDR segment field.</p> |
| &E        | S/R | <p><b>Syntax:</b> &amp;E(<i>expression</i>)</p> <p>The <i>expression</i> is evaluated and the result used as if it had been entered directly as a literal value. This keyword performs calculations without using a user exit. See “Expressions” on page 258 for more information.</p> <p>For examples of using this keyword, see “Example 7” on page 273.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

Table 39. Literal keywords (continued)

| Keyword | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &ERR    | S/R | <p><b>Syntax:</b> &amp;ERR(<i>level,code,facode,text</i>)</p> <p>Allows you to establish your own errors for a transaction. This literal keyword can be used in one of three ways:</p> <ul style="list-style-type: none"> <li>• On the Literal line itself</li> <li>• &amp;IF (expression)</li> <li>• &amp;ASSERT (expression)</li> </ul> <p><b>Note:</b> If you specify a field in the application name field, you cannot use &amp;ERR on this occurrence of the element mapping.</p> <p><i>level</i> is the severity of error where 1=data element, 2=segment, 3=transaction.</p> <p><i>code</i> is the unique error code that is associated with the error. This value can range from 0 to 999. WebSphere Data Interchange automatically adds 5000 to separate this value from WebSphere Data Interchange detected errors.</p> <p><i>facode</i> is the functional acknowledgement error code that is associated with this error (receive only).</p> <p><i>text</i> is some text that is included in an error message logged by WebSphere Data Interchange if this error is detected.</p> <p>If an &amp;ERR special literal is executed (normally controlled by either an &amp;IF or an &amp;ASSERTion), then WebSphere Data Interchange will log message TR0026. Within this message, the <i>text</i> and <i>code</i> will be identified. The <i>level</i> that you assign in the &amp;ERR becomes the extended return code from the translator and thus the JCL condition code in the WebSphere Data Interchange Utility. If <i>level</i> exceeds the acceptable error level specified in the transaction usage, then the translation will not be successful and the application data will not be returned. The value of <i>code</i> plus 5000 will also be added to the list of errors for the transaction. These are available to the API programs in the ERRCODE field of the translator control block.</p> <p>For an example of using this keyword, see “Example 8” on page 274.</p> |
| &FORCE  | R   | <p><b>Syntax:</b> &amp;FORCE value</p> <p>Forces a literal value into an application field regardless of the EDI standard data element's contents. A literal value specified for Receive mapping is normally used only if the EDI standard data element does not contain any data, or if an error occurs while processing the data (see “Using literals for Receive mapping” on page 244).</p> <p>For the &amp;FORCE keyword to be effective, you must use it in a data element of a segment that is present in the input transaction.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| &FORMAT | S/R | <p><b>Syntax:</b> &amp;FORMAT</p> <p>Substitutes the data format ID.</p> <p>You can also use the &amp;FORMAT keyword with the &amp;IFDATA, &amp;IFNODATA, or &amp;FORCE keywords.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## Using literals

Table 39. Literal keywords (continued)

| Keyword   | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &IF       | S/R | <p><b>Syntax:</b> &amp;IF(<i>expression</i>)</p> <p>The action associated with this literal is only executed if the <i>expression</i> is true. The &amp;IF literal can be combined with:</p> <ul style="list-style-type: none"> <li>• A mapping between application fields and EDI standard data elements</li> <li>• A request to SET/SAVE/USE a named variable</li> <li>• An error condition (for example, an &amp;IF followed by &amp;ERR)</li> </ul> <p>The differences between &amp;ASSERT and &amp;IF are:</p> <ul style="list-style-type: none"> <li>• &amp;ASSERT is a statement about the transaction that is expected to be true. If the &amp;ASSERT is not true, special action takes place.</li> </ul> <p>With &amp;IF, if the expression is true, a special action takes place. An &amp;IF is usually associated with a mapping or named variable, but this is not required.</p> <ul style="list-style-type: none"> <li>• &amp;IF conditions are always checked.</li> </ul> <p>See “Expressions” on page 258 for more information. For examples of using this keyword, see “Example 5a” on page 271, “Example 5b” on page 272, and “Example 5a” on page 271.</p>                                                                                                                                                    |
| &IFDATA   | S/R | <p><b>Syntax:</b> &amp;IFDATA value</p> <p>For sending, uses a literal value only if an application field contains data. For example, a segment has a pair of data elements for a qualified value and its qualifier. The application data has a corresponding field for the qualified value, but not for the qualifier. You want to supply the qualifier with a literal, but only when the application supplies a qualified value. You can do this using the &amp;IFDATA keyword with the Application field name and Literal fields on the Map Data Element panel (TP10). &amp;IFDATA can be used in combination with &amp;SET or &amp;SAVE keywords.</p> <p>The test performed by the translator to determine whether or not a source field contains data is different depending on the data type of the source field. For numeric data types, zeros are not considered significant, in which case the assumption is made that the field does not contain data. &amp;IFDATA can be used in combination with &amp;ZEROSIG if you want a zero value to be considered significant. &amp;ZEROSIG must precede the save keyword.</p> <p>&amp;IFDATA can be used on receive but only when used in combination with the &amp;SET or &amp;SAVE keywords.</p> <p>For an example of using this keyword, see “Example 2” on page 270.</p> |
| &IFNODATA | S/R | <p><b>Syntax:</b> &amp;IFNODATA value</p> <p>Uses a literal value only if an application field does not contain data. If you do not use a keyword before the literal, &amp;IFNODATA is used by default.</p> <p>&amp;IFNODATA can be used in receive transactions, but only when used in combination with the &amp;SET keyword.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

Table 39. Literal keywords (continued)

| Keyword    | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &IFNOVAR   | S/R | <p><b>Syntax:</b> &amp;IFNOVAR</p> <p>This keyword is only valid when used with &amp;SAVE, &amp;LSAVE, &amp;SET, or &amp;LSET:</p> <pre>&amp;IFNOVAR &amp;SAVE variable &amp;IFNOVAR &amp;LSAVE variable &amp;IFNOVAR &amp;SET variable &amp;IFNOVAR &amp;LSET variable</pre> <p>The named variable will only be created if it does not already exist. If the named variable already exists, the data in it is not overlaid.</p> <p>For an example of using this keyword, see “Example 3” on page 270</p>                                                                          |
| &LOOPBREAK | S   | <p><b>Syntax:</b> &amp;LOOPBREAK</p> <p>Use when an outer and an inner loop are qualified on the same record, and when you want the inner loop to be generated more than once. By putting a &amp;LOOPBREAK in the inner loop, then the inner loop will be repeated until the &amp;LOOPBREAK condition is met. The &amp;LOOPBREAK condition is established using &amp;IF; for example,</p> <pre>&amp;IF((A &lt; B) AND (X &gt; Y)) &amp;LOOPBREAK</pre> <p><b>Note:</b> You cannot enter an application field name on the same mapping occurrence that uses &amp;LOOPBREAK.</p>     |
| &LOOPCHECK | S   | <p><b>Syntax:</b> &amp;LOOPCHECK</p> <p>This keyword is very similar to &amp;LOOPBREAK, but &amp;LOOPCHECK does all the conditional processing automatically, based on the application field name used in the mapping that specifies &amp;LOOPCHECK. WebSphere Data Interchange will save this field value the first time the loop is created, and will continue to create inner loops until a record with a different non-blank field value is found.</p> <p><b>Note:</b> You cannot enter an application field name on the same mapping occurrence that uses &amp;LOOPCHECK.</p> |

## Using literals

Table 39. Literal keywords (continued)

| Keyword | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &LSAVE  | S/R | <p><b>Syntax:</b> &amp;LSAVE <i>variable</i>&lt;,&lt;<i>position</i>&gt;&lt;,&lt;<i>length</i>&gt; &lt;<i>default value</i>&gt;</p> <p>Saves the value from the current data element in a named variable, but only for the duration of the loop instance. A value saved in an outer loop is available within associated inner loops. A repeating segment is considered to be a loop; therefore, values saved in one instance of a repeating segment are not available in subsequent segment iterations.</p> <p>The value to be saved is normalized before being stored in the variable. The normalization that takes place depends on the data type of the source field being saved. For character data types A, AC, AN, CH, or ID, any trailing blanks are removed from the source value. If the value in the field (defined with a character data type) contains all numbers, the data will be treated as a numeric data type. Leading zeros will be removed from numeric data types. For numeric data types, the source value has all leading and trailing blanks removed, it is converted to a REAL value, and all leading zeros before the decimal and trailing zeros after the decimal are removed. If a source field is numeric and contains a zero value, the variable will contain null. If a value of zero needs to be saved, &amp;ZEROSIG must be used in combination with the save. &amp;ZEROSIG must precede the &amp;LSAVE keyword.</p> <p><i>position</i> is optional, but if provided, it indicates the position within the current variable where the information is saved. You can use an asterisk (*) to save at the end of the variable; for example, &amp;LSAVE name,*. If <i>position</i> is not specified, the data replaces the current value of the variable.</p> <p><i>length</i> is optional, but if provided, it indicates the length of data that is saved. You can use an asterisk to save the total length of the data being supplied; for example, &amp;LSAVE name,*,*.</p> <p>For example, if you want data to be saved starting in position 4 and the data you want saved is 5 characters long, the position and length would be stated as 4,5. Stating both the position and length, you can combine more than one data element into a single named variable. Special processing might be needed when combining data elements with different data types into a single named variable. See “Example 9” on page 275 and “Example 10” on page 276</p> <p>The <i>default value</i> is also optional, and if provided, the value is saved in the <i>variable</i> when the current data element contains no value.</p> <p>If the named variable does not already exist, it is created. If it already exists, the data in the existing variable is overlaid with the new data. If the data element is empty and no default value is supplied, the named variable is created but it contains no data.</p> <p>A variable established with &amp;LSAVE does not affect the value of a variable with the same name at any other looping level. If the variable was created outside the loop with LSAVE or SAVE, and the variable is created again inside the loop, the USE for the variable inside the loop will use the value which was saved inside the loop. If the USE for the variable is outside the loop, then the value which was saved outside the loop will be used.</p> <p>For an example of using this keyword, see “Example 3” on page 270.</p> |

Table 39. Literal keywords (continued)

| Keyword | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &LSET   | S/R | <p><b>Syntax:</b> &amp;LSET <i>variable</i>&lt;,&lt;<i>position</i>&gt;&lt;,&lt;<i>length</i>&gt; <i>value</i> The named variable is created or overlaid with the stated value, but only for the duration of the loop instance. A value set in an outer loop is available within associated inner loops. A repeating segment is considered to be a loop; therefore, values set in one instance of a repeating segment are not available in subsequent segment iterations. <i>position</i> is optional, but if provided, it indicates the position within the current variable where <i>value</i> is set. You can use an asterisk to set <i>value</i> at the end of the variable, for example, &amp;LSET <i>name</i>,* <i>abcd</i>. If <i>position</i> is not specified, <i>value</i> replaces the current value of the variable.</p> <p><i>length</i> is optional, but if provided, it indicates the length of <i>value</i> that is set. You can use an asterisk to set the total length of <i>value</i>, for example, &amp;LSET <i>name</i>,*,* <i>abcd</i>. Special processing might be needed when combining data elements with different data types into a single named variable. See “Example 9” on page 275 and “Example 10” on page 276</p> <p>A variable established with &amp;LSET does not affect the value of a variable with the same name at any other looping level. If the variable was created outside the loop with LSET or SET and the variable is created again inside the loop, the USE for the variable inside the loop will use the value which was set inside the loop. If the USE for the variable is outside the loop, then the value which was set outside the loop will be used.</p> <p>When zeros are passed in a DT application field and mapped to a DT element, they are considered significant data, and are populated to the element. In other words, WebSphere Data Interchange treats DT data as character versus numeric.</p> |
| &LSID   | R   | <p><b>Syntax:</b> &amp;LSID <i>value</i></p> <p>Identifies the instance of the LS loop, where <i>value</i> is equal to the LS01 value in the EDI standard data. This special literal is required only if the translator has no other way to determine which LS loop is provided.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| &SAMEAS | S/R | <p><b>Syntax:</b> &amp;SAMEAS <i>seqno</i></p> <p>Indicates when mapping for a current data element must be exactly the same as the data element identified by &lt;<i>seqno</i>&gt;.</p> <p>For example, if the mapping for element 2 of the POC segment must be exactly the same as the mapping for element 1 of the POC segment, then specify the special literal value of '&amp;SAMEAS 1' when mapping element 2. This results in the mapping for element 2 being the same as element 1, which is useful when qualifying (Q/S) data elements on a Receive mapping.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

## Using literals

Table 39. Literal keywords (continued)

| Keyword | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &SAVE   | S/R | <p><b>Syntax:</b> &amp;SAVE <i>variable</i>&lt;,&lt;<i>position</i>&gt;&lt;,&lt;<i>length</i>&gt; &lt;<i>default value</i>&gt;</p> <p>Saves the value from the current data element (inbound) or application field (outbound) in a named variable. The value to be saved is normalized before being stored in the variable. The normalization that takes place depends on the data type of the source field being saved. For character data types A, AC, AN, CH, or ID, any trailing blanks are removed from the source value. For numeric data types, the source value has all leading and trailing blanks removed, it is converted to a REAL value, and all leading zeros before the decimal and trailing zeros after the decimal are removed. If a source field is numeric and contains a zero value, the variable will contain null. If a value of zero needs to be saved, &amp;ZEROSIG must be used in combination with the save. &amp;ZEROSIG must precede the &amp;SAVE keyword.</p> <p><i>position</i> is optional, but if provided, it indicates the position within the current variable where the information is saved. You can use an asterisk to save at the end of the variable; for example, &amp;SAVE name,*. If <i>position</i> is not specified, the data replaces the current value of the variable.</p> <p><i>length</i> is optional, but if provided, it indicates the length of data that is saved. You can use an asterisk to save the total length of the data being supplied, for example, &amp;SAVE name,*,*.</p> <p>For example, if you want data to be saved starting in position 4 and the data you want saved is 5 characters long, the position and length would be stated as 4,5. Stating both the position and length, you can combine more than one data element into a single named variable. Special processing might be needed when combining data elements with different data types into a single named variable. See “Example 9” on page 275 and “Example 10” on page 276.</p> <p>The <i>default value</i> is also optional, and if provided, the value is saved in the <i>variable</i> when the current data element contains no value.</p> <p>If the named variable does not already exist, it is created. If it already exists, the data in the existing variable is overlaid with the new data. If the data element is empty and no default value is supplied, the named variable is created but it contains no data.</p> <p>For examples of using this keyword, see “Example 1” on page 270.</p> <p>When zeros are passed in a DT application field and mapped to a DT element, they are considered significant data, and are populated to the element. In other words, WebSphere Data Interchange treats DT data as character versus numeric.</p> |



Table 39. Literal keywords (continued)

| Keyword  | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &SET     | S/R | <p><b>Syntax:</b> &amp;SET variable&lt; ,position&gt;&lt;,length&gt; value</p> <p>The named variable is created or overlaid with the stated value. If no default value is specified, &amp;SET clears the variable.</p> <p><i>position</i> is optional, but if provided, it indicates the position within the current variable where value is set. You can use an asterisk to set value at the end of the variable, for example, &amp;SET name,* abcd. If no default value is specified, &amp;SET clears the variable.</p> <p><i>length</i> is optional, but if provided, it indicates the length of value that is set. You can use an asterisk to set the total length of value; for example, &amp;SET name,*,* abcd. Special processing might be needed when combining data elements with different data types into a single named variable. See “Example 9” on page 275 and “Example 10” on page 276</p> <p>When zeros are passed in a DT application field and mapped to a DT element, they are considered significant data, and are populated to the element. In other words, WebSphere Data Interchange treats DT data as character versus numeric.</p> |
| &THANDLE | R   | <p><b>Syntax:</b> &amp;THANDLE</p> <p>Substitutes the WebSphere Data Interchange archive key. Can be used to assist in mapping the SAP IDOC. It enables mapping of the WebSphere Data Interchange archive key to the SAP IDOC for inbound processing. The length of the THANDLE field is 20 characters and is formatted as YYYYMMDDHHMMSSnnnnnn. It is the concatenation of the date, time, and a sequence number to ensure uniqueness.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| &TIME    | S/R | <p><b>Syntax:</b> &amp;TIME</p> <p>Substitutes the system time. The length of the time field in the EDI standard data (send) or application data (receive) determines whether the time is formatted as hhmm or hhmmss.</p> <p>You can use the &amp;TIME keyword as source data for any of the EDI standard data types.</p> <p>You can also combine the &amp;TIME keyword with the &amp;IFDATA, &amp;IFNODATA, or &amp;FORCE keywords.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| &TPID    | S/R | <p><b>Syntax:</b> &amp;TPID</p> <p>Substitutes the value of the internal trading partner ID.</p> <p>You can also use &amp;TPID keyword with the &amp;IFDATA, &amp;IFNODATA, or &amp;FORCE keywords.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| &TPNICKN | S/R | <p><b>Syntax:</b> &amp;TPNICKN</p> <p>Substitutes the value of the trading partner nickname.</p> <p>You can also use &amp;TPID keyword with the &amp;IFDATA, &amp;IFNODATA, or &amp;FORCE keywords.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

## Using literals

Table 39. Literal keywords (continued)

| Keyword  | S/R | Keyword description and syntax                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &USE     | S/R | <p><b>Syntax:</b> &amp;USE <i>variable</i>&lt;,<i>position</i>&gt;&lt;,<i>length</i>&gt; &lt;<i>default value</i>&gt;</p> <p>For inbound transactions, the value of the named variable is the source of data for the application field. The value of the EDI standard data element being mapped is ignored.</p> <p>For outbound transactions, the value of the named variable is used to provide data for the EDI standard data element being mapped. An application field cannot be specified.</p> <p>The named variable must be saved or set using the appropriate keyword before you can use it with this keyword.</p> <p><i>position</i> is optional, but if provided, it indicates the position within the variable from which the data is retrieved.</p> <p><i>length</i> is optional, but if provided, it indicates the length of data that is retrieved. You can use an asterisk to move all data beginning at the location specified by the position parameter through the end of the variable, for example, &amp;USE var 3,*.</p> <p><i>default value</i> is optional, and if provided, the default value is used when the variable contains no value.</p> <p><b>Note:</b> For numeric elements, a variable value of zero causes WebSphere Data Interchange to use the default value. In the case where no default is specified, as in <code>literal = &amp;USE X</code>, there will be no output if variable X contains zero. At times, however, zero needs to be considered significant. In these cases, the user specifies the default, as in <code>literal = &amp;USE X 0</code>.</p> <p>For an example of using this keyword, see “Example 1” on page 270.</p> |
| &ZEROSIG | S   | <p><b>Syntax:</b> &amp;ZEROSIG &lt;<i>default value</i>&gt;</p> <p>Use the keyword &amp;ZEROSIG to indicate that a zero in an application field is significant when mapping to optional or conditional data elements. Without use of this special literal, WebSphere Data Interchange considers a zero value being mapped to an optional or conditional data element as insignificant and produces no output during translation. You can combine the use of &amp;ZEROSIG with &amp;IFDATA, &amp;IFNODATA, &amp;SAVE, and &amp;LSAVE keywords. &amp;ZEROSIG must precede the other keyword. &amp;ZEROSIG can also be used with a default literal to indicate that the application field is used if it contains a value, including zero, but the default literal is used if the field is blank. Binary data types preclude this because blanks represent real values.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

---

## Named variables

For receiving transactions, one way to use data in one data element for multiple application fields is to map the data element to each application field, using the repeat mapping capability. The disadvantage is that the application structures are always created, whether or not they are needed. Sometimes, you do not want the structure to be created unless some other data within the transaction is present. *Named variables* let you save the value of a data element until the time when the application structure is created. Then when you are mapping the data element that creates the structure, you can use the repeat mapping capability to map the value in the named variable.

Named variables are also critical to the use of expressions and conditional processing. See “Expressions” on page 258 and “Conditional processing of literals” on page 245 for additional information. Data values from an application field for outbound processing or standard data elements for inbound processing are not directly available for use in an expression. The values must first be saved to a named variable using for example, the &SAVE literal, then the named variable can be used within the expression.

A variable name can be the same as your application field name, up to 16 characters, but it cannot start with any of the following:

- A numeric digit (0 through 9).
- The letter P. These variables are reserved for future use.
- The letters DI. These variables are reserved for WebSphere Data Interchange.
- An ampersand (&), so they do not get confused with special literals.
- A left parenthesis, so they do not get confused with the start of an expression.

A variable name cannot contain any of the special characters designated as Arithmetic Operators or Alternate Comparison Operators. Use of these special characters within a variable name will produce unpredictable results.

The first character of a variable name determines the life span or scope of the variable:

- If the first letter is anything other than G, the variable has transaction scope. The variable is deleted after the transaction is translated. This is the same scope as local accumulators (T0 through T9). For more information, see “Using accumulators” on page 241.
- If the first letter is G, the variable has translator session scope. The variable is not deleted until the session with the translator is terminated. This is the same scope as global accumulators (G0 through G9). For more information about accumulators, see “Using accumulators” on page 241.
- To create a variable that will exist only for the duration of the loop in which it was created, use the &LSAVE or &LSET literal keywords. When the loop repeats or terminates, any variable created using these keywords is deleted. A variable established with &LSAVE or &LSET does not disturb the value of a variable with the same name at any other looping level.

Variable names are not case sensitive. *TOTALITEMS* and *totalitems* are the same variable. Special processing might be needed when combining data elements with different data types into a single named variable. See “Example 9” on page 275 and “Example 10” on page 276.

You can combine the substring capability in mapping with the substring capability of named variables. For example, you can use sub strings to get the first 4 bytes of a standard data element, then use the *position* or *length* options to put the data in a specific place in the named variable.

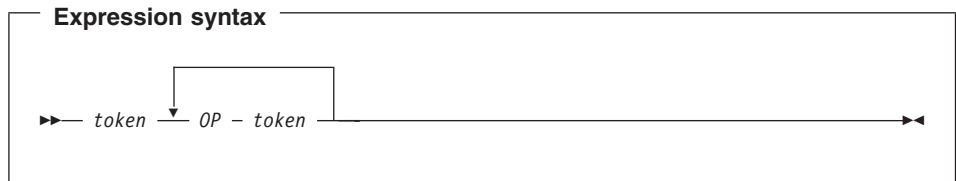
---

### Expressions

Special literals that use expressions are:

- &IF
- &ASSERT
- &E

The syntax for an expression is:



Where *token* can be any of:

- A named variable, such as TOTALS
- A numeric constant, such as 1024
- A text constant, such as 'ABCD'

**Note:** You must use quotation marks around text constants to distinguish them from named variables. You can use single or double quotation marks (' or "), but whichever is used to start the text constant must also be used to end it.

and where *OP* is one of the following types of operators:

- Boolean
- Comparison
- Arithmetic
- Unary
- Special

By default, the data type of a variable is implicitly assigned. If the contents are all numeric digits, the data type is assumed to be numeric and is treated as such in subsequent comparisons. This assignment does not take into consideration the data type of the source data field that originally supplied the value for the variable. It is based on the contents of the variable. If a variable contains any nonnumeric characters, the variable data type is assumed to be character. You can override this implicit data type assignment with the use of the CHAR or NUMBER operators which work as follows:

**CHAR** The CHAR operator forces WebSphere Data Interchange to treat a value as a character value rather than a numeric value.

**NUMBER**

The NUMBER operator forces WebSphere Data Interchange to treat a value

as a numeric value rather than a character value. WebSphere Data Interchange will normally treat a value in quotation marks as a character value and, if both operands look like character values then what might be thought of as a numeric operator will be treated as a string operator. Thus, the expression `&E('12' + '34')` will yield '1234' because both operators are flagged as character data. If what you really wanted was an arithmetic addition rather than a string concatenation then you can use `&E(NUMBER('12') + NUMBER('34'))` which will yield 46 (as would `&E(12 + 34)`).>

## Boolean operators

WebSphere Data Interchange has two Boolean operators: AND and OR. AND returns a value of 1 if both conditions are true, or a value of 0 if either condition is not true. OR returns a value of 1 if either condition is true, or a value of 0 if both conditions are not true. AND and OR must be entered in uppercase because they are case sensitive.

## Comparison operators

Comparison operators tell WebSphere Data Interchange to compare two objects.

WebSphere Data Interchange processes comparison operators the same way it does Boolean operators. If the comparison is true, it returns a value of 1. If it is not true, it returns a value of 0.

WebSphere Data Interchange has the following comparison operators (you can use either the letters or the symbols):

**EQ (=)** The first value is the same as the second value.

**GE (>=)**  
The first value is greater than or equal to the second value.

**GT (>)** The first value is greater than the second value.

**LE (<=)**  
The first value is less than or equal to the second value.

**LT (<)** The first value is less than the second value.

**NE (!=)** The two values are not equal (including being of different data types).

### Notes:

1. If you use the alphabetic operators, such as EQ, they must be uppercase.
2. Comparing a numeric constant or variable to a character constant or variable results in an invalid comparison. This produces a false result for all comparisons with the exception of Not Equal (NE or !=), in which case a true result is returned.
3. If you enclose a numeric constant in quotation marks ("), a numeric data type is still assumed for the data. Therefore, comparison to a variable that contains numeric data will not fail.

## Arithmetic operators

In the following descriptions, single quotation marks (') surround absolute values. WebSphere Data Interchange has the following arithmetic operators:

## Operators

- +**  $a + b$  has a value equal to the sum of the two. For numeric values, the sum is numeric. For example,  $2 + 2 = 4$ . For nonnumeric values, the sum is a concatenation of the values. For example, 'AB' + 'CD' equals 'ABCD'. The use of quotation marks around a numeric constant has special meaning when used during addition. The enclosed value will be treated as character data and concatenated to the second value. For example, '19' + 940323 = 19940323.
- $a - b$  has a value equal to the difference between the two. For numeric values, the result is numeric. For example  $4 - 2 = 2$ . For nonnumeric values, the result is a deconcatenation of the two values. For example, 'ABCD' - 'CD' equals 'AB'. The use of quotation marks around a numeric constant has special meaning when used during subtraction. The enclosed value will be treated as character data and deconcatenated from the second value. For example, 19940323 - '23' = 199403.
- \***  $a * b$  has a value equal to the multiplication of  $a$  and  $b$ , if both  $a$  and  $b$  are numeric. For example,  $4 * 4 = 16$ . If either variable is nonnumeric, the result is created by concatenating the value of the  $b$  after each character in  $a$ . For example, 'ABCD'\*'Z' equals 'AZBZCZDZ'.
- /**  $a / b$  has a value equal to the division of  $a$  by  $b$ , if both  $a$  and  $b$  are numeric. For example,  $16 / 4 = 4$ . Division by 0 yields a 0 value. If either variable is nonnumeric, the result is created by removing each occurrence of  $b$  from  $a$ . For example, 'ACDBCD'/'CD' equals 'AB'.

**CHAR** The CHAR operator forces WebSphere Data Interchange to treat a value as a character value rather than a numeric value.

**NOT** An operator to reverse the Boolean value of an expression. The exclamation point (!) is a short hand for the NOT operator. NOT(value) has the following meanings:

1. If value is numeric, then NOT(value) is 1 if the value is 0 and 1 otherwise. Thus, NOT(0) yields 1 and NOT(1) yields 0.
2. If value is a string value, then NOT(value) is 1 if the string has no length and zero otherwise. Thus, NOT('abc') yields 0 and NOT('') yields 1.

### NUMBER

The NUMBER operator forces WebSphere Data Interchange to treat a value as a numeric value rather than a character value. WebSphere Data Interchange will normally treat a value in quotation marks as a character value and, if both operands look like character values then what might be thought of as a numeric operator will be treated as a string operator. Thus, the expression &E('12' + '34') will yield '1234' because both operators are flagged as character data. If what you really wanted was an arithmetic addition rather than a string concatenation then you can use &E(NUMBER('12') + NUMBER('34')) which will yield 46 (as would &E(12 + 34)).

### RD or:

$a$  RD  $n$  has a value equal to  $a$  rounded to  $n$  decimal places if  $a$  is numeric. For example, 4321.556 RD 2 equals 4321.56. If  $a$  is nonnumeric, the result is created by taking the first  $n$  characters from  $a$ . For example, 'ACDBCD' RD 3 equals 'ACD'. If  $n$  is less than or equal to zero it is interpreted to be a request

to remove leading blanks. Thus, ' ABC' RD 0, yields 'ABC'. If using the character string RD for rounding instead of the special character `;`, it must be entered using uppercase.

**TU or;**  $a$  TU  $n$  has a value equal to  $a$  truncated to  $n$  decimal places if  $a$  is numeric. For example, 4321.556 TU 2 is 4321.55. Notice that the number 6 is dropped from the result and not rounded. If  $a$  is nonnumeric, the result is created by taking the last  $n$  characters from  $a$ . For example, 'ACDBCD' TU 3 yields 'BCD'. If  $n$  is less than or equal to zero it is interpreted to be a request to remove trailing blanks. Thus, 'ABCbbb' TU 0 yields 'ABC'. If using the character string TU instead of the special character `;`, it must be entered using uppercase.

## Unary operator

WebSphere Data Interchange has one unary (single component) operator, “-”.

$\&E(-a)$  changes the sign of  $a$ . If  $a$  does not exist, the value is 0.

## Special operators

**UE**  $\&E(a$  UE 'MYPROG') returns a value from the user-written program MYPROG.

**Note:** As mentioned under the descriptions of “&LSAVE” on page 252 and “&LSET” on page 253, variables are normalized to REAL value equivalents. When using this special operator, the variable value that is passed is converted back to its  $Nn$  numeric format.

**TS**  $\&E(a$  TS 'MYTABL') translates the local value  $a$  to the standard value using the MYTABL translation table.

**Notes:**

1. As noted under the descriptions of “&LSAVE” on page 252 and “&LSET” on page 253, variables are normalized to REAL value equivalents. When using this special operator, the variable value that is passed is converted back to its  $Nn$  numeric format.
2. If the translation table specified does not exist, or the value passed does not match an entry in the table, then no data is produced from this instruction and no errors or exceptions are issued.

**TL**  $\&E(a$  TL 'MYTABL') translates the standard value  $a$  to the local value using the MYTABL translation table.

**Notes:**

1. As noted under the descriptions of “&LSAVE” on page 252 and “&LSET” on page 253, variables are normalized to REAL value equivalents. When using this special operator, the variable value that is passed is converted back to its  $Nn$  numeric format.
2. If the translation table specified does not exist, or the value passed does not match an entry in the table, then no data is produced from this instruction and no errors or exceptions are issued.

**IN**  $\&E(a$  IN 'MYTABL') returns a value equal to 1 if  $a$  exists in the MYTABL validation table.

### Notes:

1. As noted under the descriptions of “&LSAVE” on page 252 and “&LSET” on page 253, variables are normalized to REAL value equivalents. When using this special operator, the variable value that is passed is converted back to its *Nn* numeric format.
2. If the validation table specified does not exist, or the value passed does not match an entry in the table, then no data is produced from this instruction and no errors or exceptions are issued.

**SC** &E(*a* SC *max.dec*) scales a real number to a maximum of *max* digits with a maximum of *dec* decimal places, truncating unused digits. For example:

```
&E(1234.56 SC 4.2) gives 1234
&E(1234.56 SC 6.2) gives 1234.56
&E(1234.56 SC 6.1) gives 1234.5
&E(1234.56 SC 8.1) gives 1234.5
&E(1234.56 SC 8.8) gives 1234.56
&E(1234.56 SC 3.3) gives 1234.56
&E(1234.56 SC 4.5) gives 1234
```

- If *max* is less than the number of significant digits to the left of the decimal point, the SC is ignored.
- If *dec* is greater than *max*, *dec* is set equal to *max*.

SC applied to strings provide a substring capability. When applied to strings, the format is &E('STRING' SC *pos.len*). For example:

```
&E('ABCDEF' SC 4.2) gives 'DE'
&E('ABCDEF' SC 1.5) gives 'ABCDE'
&E('ABCDEF' SC 9.1) gives ''
&E('ABCDEF' SC 1.9) gives 'ABCDEF'
&E('ABCDEF' SC .1) gives 'A'
&E('ABCDEF' SC 5) gives 'E'
```

If you do not provide *pos* or *len*, WebSphere Data Interchange uses 1 for that value.

**SR** &E(*a* SR *max.dec*) scales a real number to a maximum of *max* digits with a maximum of *dec* decimal places, rounding the value. For example:

```
&E(1234.56 SR 4.2) gives 1235
&E(1234.56 SR 6.2) gives 1234.56
&E(1234.56 SR 6.1) gives 1234.6
&E(1234.56 SR 8.1) gives 1234.6
&E(1234.54 SR 8.1) gives 1234.5
```

**IS** &E(*a* IS *pattern*) has a value equal to *a* but establishes a *pattern* for the data within *a*. This *pattern* only has meaning when using the TO operator (next). All variables have a default pattern of "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

**TO** &E(*a* TO *pattern*) has a value that is created by matching the pattern associated with *a* with the *pattern* in this expression. A character from TO *pattern*, if located in the IS *pattern* and if found the corresponding character from *a*, is moved to the result field. If a match cannot be found, the character from the TO *pattern* is moved to the result field. For example:

- To reverse a string:



```
&E('PLEH' IS 'ABCD' TO 'DCBA') gives 'HELP'
```

- To insert delimiters:

```
&E('HHMM' IS 'ABCD' TO 'AB:CD') gives 'HH:MM'
```

- To remove delimiters:

```
&E('HH:MM' IS 'ABCDE' TO 'ABDE') gives 'HHMM'
```

The last three examples all show the use of both the IS and TO operators for clarity. The IS operator is not really necessary because all variables automatically have the default pattern described in the IS operator. Therefore:

```
&E('PLEH' TO 'DCBA') gives 'HELP'
```

```
&E('THISAMEG ' TO 'ABCDICDIEIFGDDEHG') gives 'THIS
IS A MESSAGE'
```

## Date conversion special operators

WebSphere Data Interchange can perform any-to-any date conversions. The format of the any-to-any date conversion operator is:

```
&E(variable FD mask TD mask)
```

where:

*variable*

The value to be converted.

**FD** From Date operator. This signals that the following token is the mask that describes the date format in *variable*.

**TD** To Date operator. This signals that the following token is the mask that describes the date format that is wanted.

*mask* The mask that describes the FROM or TO date format. A mask consists of symbols which identify the date provided or date wanted. Symbols can be in upper or lower case. Any value in the mask that is not one of the symbols listed is expected to be physically part of the source data (FD) or will become physically part of the result data (TD).

**CC** Century

**YY** Year

**MM** Month of year

**DD** Day of month

**D** Day of month as a single character, if possible

**HH** Hour of day

**MM** Minute of hour

**II** Minute of hour

MM can be used when it immediately follows HH as in HHMM; however, if you want minute followed by hour, you must use IIHH, because MMHH would be interpreted as month of year and Hour of day.

## Operators

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SS</b>  | Second of minute                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>WW</b>  | Week of Year (1 through 52)                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>K</b>   | Day of Week (Monday=1, Tuesday=2, and so on)<br><br>D can be used if it immediately follows WW as in WWD; however, if you want day of week followed by week, you must use KWW, because DWW would be interpreted to be day of month and Week of year.                                                                                                                                                                                                                  |
| <b>JJJ</b> | Julian day of year                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Q</b>   | Quarter (1,2,3,4)                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>E</b>   | Semester                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>ZZZ</b> | Time zone                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>TM</b>  | Textual month (such as January, February, and so on)<br><br>TM can be followed by the name of a translate table to convert a textual month to a numeric month (FD), or from a numeric month to a textual month (FD). If a table name is not provided, the default table names are DIMONTEXT to translate from text to numeric and DIMONNUM to translate from numeric to text. A table name is indicated using parentheses, for example:<br><br>TM( <i>tablename</i> ) |

where *tablename* must be a constant.

After processing the From Date and before creating the To Date, the following processing will be done.

1. SS (seconds) will be defaulted to 0.
2. CC (century) will be defaulted to 19 when the YY (year) is greater than 10 and to 20 otherwise.
3. JJJ (julian day) will be created based on WW (week of year) and K (day of week) if not otherwise provided and WW and K were provided.
4. JJJ (julian day) will be created based on MM (month of year) and DD (day of month) if not otherwise provided and MM and DD were provided.
5. JJJ (julian day) will be used to determine WW (week of year), K (day of week), if either WW or K was not provided.
6. JJJ (julian day) will be used to determine MM (month of year), DD (day of month), if either MM or DD was not provided.
7. Q (Quarter) will be determined based on MM (month of year) if not otherwise provided.
8. E (Semester) will be determined based on MM (month of year) if not otherwise provided.

Here are some examples using CONSTANTS for all values:

- Simple example to remove delimiters.

- ```
&E('96/06/07' FD 'YY/MM/DD' TD 'YYMMDD') yields '960607'
```
- Simple example to remove delimiters and rearrange.
 - ```
&E('96/06/07' FD 'YY/MM/DD' TD 'MMDDYY') yields '060796'
```
  - ```
&E('96/06/07 EDT' FD 'YY/MM/DD ZZZ' TD 'ZZZ MMDDYY') yields 'EDT 060796'
```
- Change delimiters and convert from one form to another.
 - ```
&E('96/06/07' FD 'YY/MM/DD' TD 'YY:JJJ') yields '96:157'
```
- Textual month to Numeric month.
  - ```
&E('June 7, 1996' FD 'TM D, CCYY' TD 'YYMMDD') yields '960607'
```
- Numeric month to Textual month.
 - ```
&E('060796' FD 'MMDDYY' TD 'TM D, CCYY') yields 'June 7, 1996'
```
- Numeric month to Textual month with a special translate table that uses abbreviations for the months.
  - ```
0 FD 'MMDDYY' TD 'DDTM(ABBREV)CCYY') yields '07JUN1996'
```

Newer releases of X12 and EDIFACT standards contain segments with variable date/time formats. The format is determined by a qualifier value in the segment. WebSphere Data Interchange provides two tables for dynamically translating the qualifier into a mask. Table DIXDTMSK is used for X12 and table DIEDTMSK is used for EDIFACT.

Assume the following X12 data is received in the DTP segment (Date or Time period):

- DTP**D8*19940927!
- Where D8 is the date or time period format qualifier (CCYYMMDD) and 19940927 is the date or time period.

Assume your application requires the date in YYMMDD format. You map the DTP segment as follows:

- DTP03 - &SAVE Qual
Results in Qual is D8
- DTP03 - &SAVE Date
Results in Date is 19940927
- DTP03 - &FORCE &E(Date FD (QUAL TS 'DIXDTMSK') TD 'YYMMDD')
Results of (Qual TS 'DIXDTMSK') is CCYYMMDD
Results of &FORCE is 940927

Order of precedence

During processing, all expressions are evaluated from left to right. The order of precedence is:

1. Unary minus (-)
2. Rounding (RD) and truncating (TU)
3. Special operators (UE, TS, TL, IN, IS, TO, SC, SR, FD, TD)
4. Multiply (*), Divide (/)
5. Addition (+), Subtraction (-)

Operators

6. Relational Operators (GT, GE, LT, LE, EQ, NE)
7. Boolean (AND)
8. Boolean (OR)

Precedence can be overridden with parentheses embedded within an expression. For example, $\&E(2+3*5)$ equals 17 because the multiplication is done first, then the addition. $\&E((2+3)*5)$ equals 25 because the parentheses indicate that the addition is done first, then the multiplication.

Special variables

WebSphere Data Interchange has reserved the prefix DI for variables that will be reserved for use WebSphere Data Interchange to accomplish special functions. The following DI variables are currently available and are used with the `&SET` keyword:

DIAPPPFILE

Use this variable to change the name of the file to which the translation application data will be written during a Receive Translate. It will override any value that was used in the receive usage or in the application data format definition. It provides the capability for data that is being received to influence the final destination for the data. For example, the statement

```
&SET DIAPPPFILE SPECIAL
```

would force the current transaction to be written to the application file identified by the ddname SPECIAL.

DIAPPTYPE

This variable sets the application file type that corresponds with the file name provided by DIAPPPFILE.

DIAUTOCC

Allows automatic century manipulation for both inbound and outbound translation. Century will be automatically added or removed from the date using the length of the standard data element or application field. For example, the following statement uses a value of 1:

```
&SET DIAUTOCC 1
```

DICCTRL

Use to remove the century control year from the translator and enable the selection of the century control year. If year is greater than 10, century is 19; otherwise century is 20. The century control year is 10. For example, if year is less than 95, century is 20, control year is 95:

```
&SET DICCTRL 95
```

DICUSERDATA

This variable is used to set the data value that will be inserted in the TRCB field cuserdata. This field is copied to any output 'C' record before the 'C' record is written. Received data can be placed into a named variable in any combination up to 256 bytes. Then use the reserved variable DICUSERDATA anytime the value of the named variable needs to be placed into the TRCB.

For example, suppose a named variable tvar has been created and filled with the data from a previously mapped data element.

That data can be placed in the TRCB cuserdata field by including the following in a map:

```
&SET DICUSERDATA &E(TVAR)
```

DIERRFILTER

This variable can be used to control which errors are actually meaningful to you at a point in time during a translation. A description of the error filter can be found in the *WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide*, SC34-6217-01.

DIEXPTRACE

This variable, when given a nonzero value (&SET DIEXPTRACE 1), causes WebSphere Data Interchange to create a TRACE of the results of all expression evaluations. When tracing is active, WebSphere Data Interchange will write out message TR0411 to the PRTFILE for each expression. The message will show the expression being evaluated and the result of the evaluation. Tracing will remain active until the DIEXPTRACE is given a zero value (&SET DIEXPTRACE 0).

Note: The TR0410 and TR0411 messages always occur as the first messages for a transaction. They are not merged with any other error messages for the transaction.

DIMAPCHAIN

This variable can be used when an inbound transaction is required by more than one application program. It enables more than one mapping to be executed for the specified transaction. The last value given to DIMAPCHAIN in a mapping will establish the application sender ID value that will be used to locate the next mapping to execute. For example, if MAPABC had this coded:

```
&SET DIMAPCHAIN APPLICATIONB
```

The inbound transaction would be translated using map MAPABC, and then it would be translated using the map that is associated with application sender ID APPLICATIONB. The DIMAPCHAIN command will cause all maps indicated by each DIMAPCHAIN command to be translated, whereas the DIMAPSWITCH command will stop translating the map that has the DIMAPSWITCH variable in it, and literally switch to the new map indicated in the command.

DIMAPSWITCH

This variable can be used when data being received needs to be inspected before it can be determined exactly what mapping is done against the transaction. With this variable, you can switch the map that is being executed dynamically based on the data that is being received. A map can be created to initially look at the data being received. Only those data elements necessary to make a mapping decision would be mapped. WebSphere Data Interchange would determine the real map to be used by interpreting values resulting from conditional logic expression. For example, a map would contain conditional logic expression:

special variables

```
&IF(X > Y) &SET DIMAPSWITCH APPLICATIONA
```

Here, if X is greater than Y, the mapping identified with an application sender ID value of APPLICATIONA would be used to translate the transaction.

DISAPSEQ

This variable can be used to save the SAP IDOC record sequence number on the first error encountered during outbound processing. The sequence number can be provided through the application or using the WebSphere Data Interchange accumulators. Variable DISAPSEQ is captured in the SAP status record to indicate the first record in error. For more information, see the *WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide*, SC34-6217-01.

DIVALLEVEL

This variable can be used to control the level of validation done. It can have the same values as the validation level specified in a usage record, which are: 0 (no validation), 1 (validation tables activated), and 2 (validation tables plus type checking). Any value other than 0, 1, or 2 will be treated as a 0.

DIVALTYPE

This variable can be used to control the data types for which data type checking is done (validation level of 2). The types that can be specified are DT, TM, N, R, CH, AN, A, and HX. They must be specified in uppercase and separated by a comma. Any value specified that is not valid is ignored. For example, to activate DT, TM and HX validation, the following can be done:

```
&SET DIVALTYPE DT, TM, HX
```

DIVARTRACE

This variable, when given a nonzero value (&SET DIVARTRACE 1), causes WebSphere Data Interchange to create a TRACE of all accesses to variables. When tracing is active, WebSphere Data Interchange will write out message TR0410 to the PRTFILE for each variable access. The message will indicate the variable being accessed and its current value. Tracing will remain active until the DIVARTRACE is given a zero value (&SET DIVARTRACE 0).

Mapping techniques for literal keywords

As you will see in the following examples, deciding where to save variables, execute expressions, and subsequently use variables is fundamental. To determine where these operations are done, it is essential to understand the order in which the translator executes element mapping instructions. As stated earlier, transactions are always processed starting with the first data element of the first segment and proceeding to the next element of the same segment, after which the next segment is processed, and so on. If one element has repeat mappings, the instructions are executed in a top-down fashion.

It is equally important to understand the difference between a mapping that might contribute to the output versus one that does not. For example, specifying &SAVE *variable* or &SET *variable* will not contribute to the output directly. Only direct mappings such as specifying an Application Field Name, forcing a literal value, specifying &USE

variable, or evaluating an &IF expression, can contribute to the output. Determining where to save and use variables is different for inbound and outbound translation.

For send processing, proper placement is easy for mappings that contribute to the output because you know which element requires the result. Proper placement for a mapping that does not contribute is not as obvious because there might be no data element relationship with this action. In most cases, you will save an application value into a variable, check or manipulate the variable, then use it. In other cases, it might not be appropriate to perform all these actions in successive repeat element mappings. For instance, you can have two independent looping structures (records) and you need to save a value from a particular iteration in the first loop. This particular value must be saved while the translator is processing this first loop. The saved variable can then be inspected and manipulated in the second repeating loop to provide the required result.

In summary, the location of a mapping that saves a variable can be far from a corresponding mapping that actually uses the variable. The best technique for deciding where to save values is to do so at or near processing points in the map when the translator is working on the corresponding record.

Note: Typically, the map for saving a value into a variable will precede the event of actually using the variable. A feature called &DEFERRED &USE can be utilized in the event that output is necessary in an earlier segment than where the final result will be set.

For receive processing, the situation is reversed. It is easy to decide where to save variables because you already know which element value in the input you need to work with. Proper placement for a mapping that contributes to your output application data is not as obvious because there might be no data element relationship with this action. In most cases, you will save a data element value into a variable, check or manipulate the variable, then use it in your application record. In other cases, it might not be appropriate to perform all these actions in successive repeat element mappings. For instance, you might need to save values from two different segments, compare them, and write the result to an application field. One input element that needs to be saved is in the header section of the EDI transaction and the second is in an outer loop (the name loop for example).

The output application field that needs to be output after comparing these two variables is related to the detail loop (line item loop for example). It is most appropriate to use the result within the detail loop because this might have an independent repeating record that is not associated with the name or header segments. Hence, the comparison and use of the result is done within this detail loop that controls the creation of the detail record. The best technique for deciding where to use variables is to do so at or near processing points in the map when the translator is working on the corresponding record.

Note: For receive processing, the map for saving a value into a variable must precede the event of actually inspecting or using the variable. The feature of &DEFERRED &USE cannot be used during inbound translation.

Examples of using literal keywords and named variables

Example 1

For an inbound transaction, if your trading partner sends you the city name, state abbreviation, and zip code in three fields, but your database puts all of this information in one 30-byte field, ADDRESS, you can use &SAVE to put the address information into the single field:

1. Map the city name data element using the literal &SAVE citystzip,1,19.
This creates the named variable citystzip and places the received city in the first 19 bytes of the named variable.
2. Map the state abbreviation data element using the literal &SAVE citystzip,20,2.
This places the received state abbreviation in the bytes 20 and 21 of the named variable.
3. Map the zip code data element using the literal &SAVE citystzip,22,9.
This places the received zip code in bytes 22 through 30 of the named variable.
4. Repeat the mapping of the zip code data element. Specify application field ADDRESS and the literal &USEbcitystzip.
WebSphere Data Interchange uses the value in the named variable that was concatenated in the first three steps.

Example 2

For an outbound transaction, you want to provide a telephone number, either of a specific contact (CONTACTPHONE) or of the organization (ORGPHONE). If a contact phone number is provided, you want to use it; otherwise, you want to use the organization phone number. CONTACTPHONE occurs before ORGPHONE.

1. In the first mapping of the phone number data element, specify application field CONTACTPHONE and the literal **&IFDATA &SAVE Tphone**.
This creates the named variable *Tphone* only if the CONTACTPHONE field contains data.
2. Repeat the mapping of the phone number data element. Specify application field ORGPHONE and the literal **&IFNOVAR &SAVE Tphone**.
This creates the named variable *Tphone* only if the named variable did not already exist (CONTACTPHONE did not contain any data).
3. Repeat the mapping of the phone number data element. Specify the literal **&USE Tphone**.

Example 3

For an inbound transaction, the application field NAME receives the value from EDI standard data element 123 (CUSTNAME) in the first occurrence of the name and address loop, or from the data element (ORGNAME) in the second occurrence of the loop, if the first occurrence does not contain data.

1. When mapping the first occurrence of the loop, specify the literal **&IFDATA &SAVE Name**. Do not specify an application field.
This will create the named variable Name only if the first occurrence contains data.

2. When mapping the second occurrence, specify the literal **&IF(Name EQ ") &LSAVE Name**. Do not specify an application field.

This will create the named variable Name only if it does not already exist (the first occurrence did not contain any data). &LSAVE is used so that the value of Name from the first occurrence of the loop is not disturbed.

3. Repeat the mapping for the second occurrence. Specify application field NAME and the literal **&USE Name**.

This will use either the value saved in step 2 (first occurrence did not contain a value) or the value saved in step 1 on page 270 (first occurrence did contain a value). All future occurrences of the loop and unrelated loops or segments only see the value of *Name* saved in step 1 on page 270.

Example 4

For an inbound transaction, the application field NAME receives the value from EDI standard data element 123 (CUSTNAME) in the second occurrence of the name and address loop, or from the data element (ORGNAM) in the first occurrence of the loop, if the second occurrence does not contain data.

1. When mapping the first occurrence of the loop, specify the literal **&SAVE Name**. Do not specify an application field.

This will create the named variable Name.

2. When mapping the second occurrence, specify the literal **&IFDATA &SAVE Name**. Do not specify an application field.

This will overlay the named variable Name only if the second occurrence contains data.

3. Repeat the mapping for the second occurrence. Specify application field NAME and the literal **&USE Name**.

This will use either the value saved in step 2 (second occurrence contained a value) or the value saved in step 1 (second occurrence did not contain a value).

Example 5a

For an outbound transaction, either application field ORDQTY or MINQTY supplies a value for an EDI standard data element. The field with the largest value is used. If neither ORDQTY or MINQTY contain data then a value of 100 is used.

1. In the first mapping of the data element, specify application field ORDQTY and the literal **&SAVE QTY1**.

This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

2. Repeat the mapping of the data element. Specify application field MINQTY and the literal **&SAVE QTY2**.

This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

3. Repeat the mapping of the data element. Specify the literal **&IF(QTY1 >= QTY2) &USE QTY1 100**. Do not specify an application field.

This compares the values of the named variables. If QTY1 is greater than or equal to QTY2, WebSphere Data Interchange will use QTY1. If neither ORDQTY or

Literal keywords and named variables

MINQTY contain a value then both QTY1 and QTY2 will have a value of 0 and will therefore be equal. However, a value of 0 is not significant, which causes WebSphere Data Interchange to use the default literal value of 100.

4. Repeat the mapping of data element 123. Specify the literal **&IF(QTY1 < QTY2) &USE QTY2**. Do not specify an application field.

This compares the values of the named variables. If QTY1 is less than QTY2, WebSphere Data Interchange will use QTY2.

Example 5b

Assume you have created a mapping similar to that of example 5a, but forgot to do step 1. Because it is necessary to define QTY1 before using it, you need to insert a mapping of this data element, making it the first occurrence of the loop. Instead of remapping all of the occurrences, perform the following:

1. Drag application field ORDQTY to the data element.
2. Open the Mapping Data Element editor for the new mapping.
3. Enter the literal **&SAVE QTY1** and click OK.
4. Drag the new mapping to the appropriate location and drop it there.

Example 6

Your application can generate three different discount rates: the regular discount (REGDISC), a volume discount (VOLDISC), and a special discount (SPECDISC). For an outbound transaction, if application field SPECDISC contains a value, you want to use it for EDI standard data element 456. However, if SPECDISC does not contain a value, then the larger value of either REGDISC or VOLDISC is used.

1. In the first mapping of data element 456, specify application field REGDISC and the literal **&SAVE REGDISC**.

This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

2. Repeat the mapping of data element 456. Specify application field VOLDISC and the literal **&SAVE VOLDISC**.

This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

3. Repeat the mapping of data element 456. Specify application field SPECDISC and the literal **&IF(REGDISC >= VOLDISC) &E(REGDISC)**.

This mapping will be executed only if the value of REGDISC is greater or equal to the value of VOLDISC. If this is the case, then SPECDISC will be mapped to the data element. However, if SPECDISC does not contain any data, then the default literal value of **&E(REGDISC)** will be used.

4. Repeat the mapping of data element 456. Specify application field SPECDISC and the literal **&IF(REGDISC < VOLDISC) &E(VOLDISC)**.

This mapping will be executed only if the value of REGDISC is less than the value of VOLDISC. If this is the case, then SPECDISC will be mapped to the data element. However, if SPECDISC does not contain any data, then the default literal value of **&E(VOLDISC)** will be used.

Notes® on examples 5 and 6

Examples 5 and 6 illustrate the differences between &IF, &USE, and &E:

- &IF is used to determine if a mapping is executed. The value of the &IF expression is not used in the mapping; it only controls the execution of the mapping. If the expression is true (nonzero value), the mapping is executed. If the expression is false (zero value) the expression is not executed. In examples 5 and 6, only one of the maps in steps 3 and 4 will be executed because the expressions are mutually exclusive.
- &USE indicates that a named variable is used as the primary source for data in the mapping. An application field cannot be specified, but you can have a default literal value if the variable name being used does not contain any data.
- &E is used exactly the same as a literal value, but instead of having a constant literal value, &E is computed, taken, or computed and taken from a named variable. If the last operator of an expression is a Boolean or comparison operator, then the value of the expression will either be true (1) or false (0), and you can map these values.

For example, if a data element contains a 1 if FLD1 is greater than FLD2, or a 0 if it is not, then mapping the expression &E(FLD1 > FLD2) would result in either a 1 or 0 being moved to the EDI standard data element. If a data element contains a 1 if FLD1 is greater than FLD2, or nothing if it is not, then the conditional mapping &IF(FLD1 > FLD2) 1 would map the constant value of 1 only when FLD1 is greater than FLD2.

Example 7

For an outbound transaction, data element 321 must have a value of S if the value of application field SIZE is 6, 7, or 8; a value of M if SIZE is 9, 10, 11, or 12; and a value of L if SIZE is 13, 14, or 15. These values are specified in the translation table SIZETAB. However, for this transaction, SIZE can also be less than 6, in which case data element 321 must have a value of XS, or greater than 15, in which case data element 321 must have a value of XL.

1. In the first mapping of data element 321, specify application field SIZE and the literal **&SAVE Size**.
This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.
2. Repeat the mapping of data element 321. Specify the literal **&IF(Size > 0 AND Size < 6) XS**.
This compares the value of the named variable to 6 and 0. If the value is less than 6 but greater than 0, WebSphere Data Interchange uses the value XS.
3. Repeat the mapping of data element 321. Specify the literal **&IF(Size > 15) XL**.
This compares the value of the named variable to 15. If the value is greater than 15, WebSphere Data Interchange uses the value XL.
4. Repeat the mapping of data element 321. Specify application field SIZE and the literal **&IF(Size >= 6 AND Size <= 15) &E(Size TS 'SIZETAB')**.
This compares the value of the named variable to 6 and 15. If the value is greater than or equal to 6, and less than or equal to 15, WebSphere Data Interchange uses the translation table SIZETAB to determine the corresponding value for the named variable.

Literal keywords and named variables

Example 8

Assume you have a lumber supply business and your trading partners are home builders in the area. The unit of measure on their orders ranges from inches to rods (one rod equals 5 yards). You have implemented just-in-time inventory processes so if you receive an order that requires additional inventory, your order is immediately sent to your supplier. Your application stores all measurements in board feet, and therefore must convert all incoming data to board feet.

Your application fields are QUANTITY and UNITMEAS. Using the conditional processing literals, you would:

1. Map the unit of measure data element, using the literal **&SAVE UOM**. Do not specify an application field.
This saves the present value of the standard data element in the named variable UOM.
2. Repeat the mapping of the unit of measure data element, using the literal **&IF(UOM EQ 'IN' OR UOM EQ 'BF' OR UOM EQ 'RD') &SET UOMOK 1**. Do not specify an application field.
This statement enables you to determine if you have only the values you want, and if the statement is true, WebSphere Data Interchange puts a 1 in the named variable UOMOK for later use.
3. Repeat the mapping of the unit of measure data element. Specify application field UNITMEAS and the literal **&IF (UOMOK EQ 1) &FORCE BF**.
This forces the value 'BF' into the application field only when the unit of measure is a valid value. The steps that follow will convert the data received into board feet.
4. Map the quantity data element, using the literal **&SAVE QTY**. Do not specify an application field.
This saves the present value of the standard data element in the named variable QTY.
5. Repeat the mapping of the quantity data element. Specify application field QUANTITY and the literal **&IF(UOM EQ 'IN') &FORCE &E(QTY/144)**.
This statement saves the value of quantity converted to board feet in the application field QUANTITY only if the incoming unit of measure was 'IN'.
6. Repeat the mapping of the quantity data element. Specify application field QUANTITY and the literal **&IF(UOM EQ 'RD') &FORCE &E(QTY*198/144)**.
This statement saves the value of quantity converted to board feet in the application field QUANTITY only if the incoming unit of measure was 'RD'.
7. Repeat the mapping of the quantity data element. Specify application field QUANTITY and the literal **&IF(UOM EQ 'BF') &FORCE &E(QTY)**.
This statement saves the value of quantity converted to board feet in the application field QUANTITY only if the incoming unit of measure was 'BF'.
8. Repeat the mapping of the unit of measure data element, using the literal **&ASSERT1(UOMOK EQ 1) &ERR(2,100,,'Invalid unit of measure')** Do not specify an application field.
This statement creates a translation error for anything that does not meet our criteria. Assume you received something in yards. You can add a repeat mapping for yards, then retranslate. The transaction would then pass through the translation.

An alternative to using conditional processing literals would be a single mapping using a translation table. For example, you can set up a translation table UOMDIV as follows:

Table 40. Sample translation table UOMDIV

Local value	Standard value
IN	144
RD	1.375
BF	1

If a translation table is used, then the mapping can be reduced to the following:

1. Map the unit of measure data element, using the literal **&SAVE UOM**.
2. Repeat the mapping of the unit of measure field, using the literal **&SET divisor &E(UOM TS 'UOMDIV')**.
3. Repeat the mapping of the unit of measure field, using the literal **&IF divisor NE 0) &FORCE BF**.
4. Map the quantity data element using the literal **&SAVE QTY**.
5. Repeat the mapping of the quantity data element, using the literal **&FORCE &E(QTY/divisor)**.
6. Repeat the mapping of the unit of measure data element, using the literal **&ASSERT1(divisor NE 0) &ERR(2,100, 'Invalid unit of measure')**.

This method might be preferred if the values of UOM are expected to change. If this happens, only the UOMDIV table needs to be updated rather than changing the mapping.

Example 9

For an outbound transaction, the application field TEST1 is an N2 data type and contains the value 100. You need to build a variable that contains "XX" in the first and second positions, "01" in the third, fourth, fifth, and sixth positions, and the value in the application field TEST1 beginning in the seventh position.

To put this information into a single variable, do the following:

1. Map a data element using the literal **&SET TVAR**.
This creates the named variable TVAR and sets the variable to null.
2. Repeat the data element and map the data element using the literal **&SET TVAR,*,2 XX**.
This places "XX" in bytes 1 and 2 of the variable TVAR.
3. Repeat the data element and map the data element using the literal **&SET TVAR,*,4 01**.
This places "01" in bytes 3 through 6 of the variable TVAR.
4. Repeat the data element and map the data element using the application field TEST1 and a literal **&SAVE TVAR,*,7**.
This places "100" in bytes 7 through 9 of the variable TVAR. The variable TVAR now contains the following:

Literal keywords and named variables

XX01 100

Note: At this point, the variable TVAR has been normalized to the data type of the application field TEST1, which is defined as an N2 data type.

To map TVAR position 3 through 6 to an EDI standard data element with an ID data type, you would need to do the following because of the normalization of the variable to an N2 data type:

1. Map a data element using the literal **&E(CHAR(TVAR) SC 3,4)**.

The &E computes the literal. The CHAR forces the variable TVAR to have a data type of character for this instruction. The SC is a scaling function but can be used to substring character data. The resulting value would be "01".

An alternative to using the scaling function is as follows:

1. Map a data element using the literal **&E(CHAR(TVAR) TO 'CDEF')**.

The TO 'CDEF' is used to create the value by matching the default pattern ('ABCDEFGHIJK...') with the 'CDEF' pattern in this expression. The resulting value would be "01".

2. Map a data element using the literal **&USE TVAR,3,4**.

The resulting value would be "0.01".

To map TVAR position 7 through 9 to an EDI standard data element with an R data type, you can move the data.

1. Map a data element using the literal **&USE TVAR,7,3**. The resulting value would be "1.00".

Example 10

For an outbound transaction, the application field TEST1 is an N2 data type and contains the value 123. Application field TEST2 is an N0 data type and contains the value 100.

To put this information into a single variable do the following:

1. Map a data element using the literal **&SET TVAR**.

This creates the named variable TVAR and sets the variable to null.

2. Repeat the data element and map the data element using the application field TEST1 and a literal **&SAVE TVAR,*,3**.

This places "123" in bytes 1 through 3 of the variable TVAR.

3. Repeat the data element and map the data element using the application field TEST2 and a literal **&SAVE TVAR,*,3**.

This places "100" in bytes 4 through 6 of the variable TVAR.

The variable TVAR now contains the following:

123100

Note: At this point the variable TVAR has been normalized to the data type of the application field TEST1 which is defined as an N0 data type.

To map TVAR position 1 through 3 to an EDI standard data element with an R data type, you would need to do the following because of the normalization of the variable to an N2 data type:

1. Map a data element using the literal **&SET TEMPVAR &E(CHAR(TVAR) SC 1.3)**.
The &E computes the literal. The CHAR forces the variable TVAR to have a data type of character for this instruction. The SC is a scaling function but can be used to substrng character data. The resulting value would be "123".
2. Repeat the data element and map the data element using the literal **&E(TEMPVAR / 100)**.
The resulting value would be 1.23.

An alternative is as follows:

1. Map a data element using the literal **&USE TVAR,1,3**.
The resulting value would be 123.

Control data literals

Audit and control are generally high priority items. If you have the need to add the internal trading partner nickname of your trading partner to your application data for each transaction, use the &TPID or &TPNICKN literals. If you need to include the data format ID associated with the current transaction, use the &FORMAT literal. Finally, if you require the application control value associated with the current transaction, use the &ACFIELD literal. This value will only be correct after ALL the fields that comprise the application control field have been processed.

Using service segment fields

Service segments are the segments used when an EDI Transaction is enveloped (ISA, GS, ST, UNB, UNH, UNT, and so on). Use the PERFORM command keyword SERVICESEGVAl to indicate which validation level must be applied to the service segments. If you do not specify the SERVICESEGVAl keyword, no service segment validation occurs. Valid values for the keyword are:

- 1 Validates the service segments for syntax only. This includes checking for mandatory data that is missing, as well as Data Elements that are too large or too small.
- 2 In addition to level 1 checking, level 2 checking validates the service segment date and time data elements according to their types. Also, Data Elements that specify a Code List will have the value of the Data Element validated against the values contained in the Code List.

Service segment validation occurs on EDI Standard Transactions that are used as the source of a translation. Service segment validation is not available for EDI documents being created as the result of a translation. Validation errors that result from service

Control data literals

segment validation cause the interchange, group, or Transaction with the error to be skipped (not processed). Translation continues on the remainder of the input data.

The SERVICESEGVAl keyword is used with the following PERFORM commands:

- DEENVELOPE
- DEENVELOPE AND TRANSLATE
- ENVELOPE
- ENVELOPE AND SEND
- RECEIVE AND DEENVELOPE
- REENVELOPE AND TRANSLATE
- REENVELOPE
- REENVELOPE AND SEND
- TRANSLATE AND ENVELOPE
- TRANSLATE AND SEND TRANSFORM

Mapping service segment fields (send only)

During the Send mapping process, the &TCN special literal can be used to indicate the application field which contains the message or transaction control number. Any field from a segment currently mapped can be chosen and either mapped or repeated, and the special literal &TCN used to identify the application field containing the control number.

- If the application field is part of a record that occurs more than once, the first record is the only one that will be used.
- It is possible to have more than one mapping which contributes to the transaction control number. The data from each mapping will be concatenated with the current data from previous mappings.
- If &DATE is used in the mapping, the format will be yyyyymmdd. If &TIME is used in the mapping, the format will be hhmmss.

The transaction control number is extracted at translate time so if delayed enveloping is used and &DATE or &TIME, or both, are used to construct the control number, the date and time will be the date and time of translation and not the date and time of enveloping.

- Translation tables and validation tables can be used during &TCN mappings the same as they can be used in any other mapping.
- The transaction control number generated by the application is truncated to the maximum length for a control number specified by the EDI standard, never to exceed fourteen bytes.
- It is possible to combine delayed enveloping with application assignment of control numbers. If this is done, when an envelope operation is requested, the transactions will be sorted such that all transactions for which WebSphere Data Interchange will assign control numbers will occur before the transactions for which the application has assigned control numbers. Also, during an envelope operation, a switch from a transaction that requires WebSphere Data Interchange to assign the control number to a transaction with application assigned control numbers will cause a new interchange to be started. Transactions with application assigned control numbers will be sorted by the value of the transaction control number.

Error message TR0115 is displayed if the application field containing the control number was not provided, if it contained all blanks, or was otherwise invalid.

Error message TR0116 is issued if the control number assigned is a duplicate within the group/interchange. The translator requires that message control numbers must be unique within the group. If groups are not being used, then message control numbers must be unique within the interchange.

These errors are considered level 3 errors.

You must fix the application so that the message control numbers are unique within the interchange or group.

The following table shows the type of validation done for every possible mapping between application data types (down) and EDI standard data types (across) during a TRANSLATE TO STANDARD processing.

Mapping specific service segment fields (receive only)

Table 41 lists the literals that are provided so that every field within every service segment can be accessed using a combination of the segment ID (ISA, UNB, STX, and so forth) concatenated with a 2-byte number indicating the field within the segment wanted. The names created with this concatenation match the names of the fields defined in the E, I, T, U, and X profiles.

Invalid names (for example, ISA44) are not flagged as errors, but return no data. Using names that do not match the envelope type being received (for example, using ISA01 when UN/EDIFACT service segments (UNB) are being used) is not an error, but no data is returned.

Table 41. Literals to identify fields in service segments

Literal:	nn Value:	Segment:
&ISA nn	01 through 16	ISA
&GS nn	01 through 08	GS
&ST nn	01 through 02	ST
&SE nn	01 through 02	SE
&GE nn	01 through 02	GE
&IEA nn	01 through 02	IEA
&UNB nn	01 through 18	UNB
&UNG nn	01 through 13	UNG
&UNH nn	01 through 09	UNH
&UNT nn	01 through 02	UNT
&UNE nn	01 through 02	UNE
&UNZ nn	01 through 02	UNZ
&STX nn	01 through 12	STX
&BAT nn	01 through 01	BAT

Mapping service segment fields

Table 41. Literals to identify fields in service segments (continued)

Literal:	nn Value:	Segment:
&MHDnn	01 through 06	MHD
&MTRnn	01 through 01	MTR
&EOBnn	01 through 01	EOB
&ENDnn	01 through 01	END
&BGnn	01 through 07	BG
&EGnn	01 through 04	EG
&ICSnn	01 through 10	ICS
&ICEnn	01 through 02	ICE

Mapping generic service segment fields (receive only)

You can map received envelope data to application fields by using substitution keywords in the **Literal** field. The keywords indicate which service segment field is mapped to the application field.

Table 42 describes the substitution keywords you can use to map service segment fields. The Envelope Data Type column indicates the required data type for the service segment field. The EDI Standard Data Type indicates the data type that WebSphere Data Interchange uses for conversions from an EDI standard data type to the application data type.

Table 42. Keywords for mapping envelope data (receive only)

Keyword:	Envelope Data Type:	EDI Standard Data Type:	Envelope Data Mapped to Application:
&I		A	Entire interchange service segment, up to the length of the application field
&ICN	CN or IV	AN	Interchange control number
&IIS	IS, AS, or RS	A	Interchange sender ID
&IIR	IR, AR, or RR	A	Interchange receiver ID
&IDT	DT	DT	Interchange date
&ITM	TM	TM	Interchange time
&IPW	PW	A	Interchange password
&IAP	AP	A	Interchange application reference
&IVR	VR	A	Interchange version/release
&IGT		N0	Interchange total number of groups
&ICT	CT	N0	Interchange control total from the interchange trailer segment
&ITT		N0	Interchange total number of transactions

Table 42. Keywords for mapping envelope data (receive only) (continued)

Keyword:	Envelope Data Type:	EDI Standard Data Type:	Envelope Data Mapped to Application:
&G		A	Entire group service segment, up to the length of the application field
&GCN	CN or IV	AN	Group control number
&GFG	FG	A	Functional group ID
&GAS	AS, IS, or RS	A	Group application sender ID
&GAR	AR, IR, or RR	A	Group application receiver ID
&GDT	DT	DT	Group date
>M	TM	TM	Group time
&GPW	PW	A	Group password
&GVR	VR	A	Group version
&GLV	LV	A	Group release
>T		N0	Group total number of transactions
&T		A	Entire transaction service segment, up to the length of the application field
&TCN	CN or IV	AN	Transaction control number. The &TCN keyword is valid for both send and receive. See “Mapping service segment fields (send only)” on page 278 for an explanation on the use of &TCN for application assigned control numbers.
&TTC	TC	A	Transaction code
&TVR	VR	A	Transaction version
&TLV	LV	A	Transaction release
&TTS		N0	Transaction total number of segments

WebSphere Data Interchange interprets any literal beginning with an ampersand (&) as a special keyword. To use a literal that begins with an ampersand, use two ampersands. The translator discards the first one and uses the remaining characters as literal data. For example, if you enter &T in the **Literal** field, the translator moves the entire transaction service segment to the application data. If you enter &&T in the **Literal** field, the translator removes the first & and uses &T as literal data. The service segments (ISA, GS, ST, UNB, UNG, and so on) are not provided in the list of segments that can be mapped for trading partner transactions. Because they are not provided in the list, a direct mapping of a field from a service segment is not possible. In order to use one of the literals from Table 39 on page 246 or Table 42 on page 280, you must map, or repeat map, some other data element defined in the transaction. When one of

Mapping service segment fields

the special literals is used, WebSphere Data Interchange knows that the value of the data element being mapped must be ignored, and the value of the special literal must be used instead.

Restrict the mapping of service segment fields to data elements in nonrepeating segments. Select a data element in the first nonrepeating segment that you know will always be present in the received transaction data. The segment *must* be present in the data being received for the mapping instructions to be executed. Repeat the element mapping as many times as is necessary to map (or &SAVE) all of the service segment fields you need. WebSphere Data Interchange recognizes the substitution keywords and moves the value from the associated service segment field rather than the transaction data element that is currently being mapped to the application field or named variable. Data conversions and table translations, and user exits are possible when service segment special literals are used.

Validation during mapping

Table 43 describes the type of validation used during translation, based on the type of data being validated.

Table 43. Validation used during mapping for different data types

Data type:	Validation:
A	The ALPHANUM validation table shipped with WebSphere Data Interchange, with numeric digits (0 through 9) removed, is used to validate the data.
AN	The ALPHANUM validation table shipped with WebSphere Data Interchange is used to validate the data.
BIN	A generic data type that encompasses the P, L, Z, B, I and H data types.
CH	The CHARSET validation table shipped with WebSphere Data Interchange is used to validate the data.
N	Only digits, leading or trailing sign characters, and leading or trailing blanks are valid.
R	Only digits, decimal notation, leading or trailing sign characters, and leading or trailing blanks are valid.
DT	Must be a valid date according to the format specified during the mapping process.
TM	Must be a valid time.
-	Validation is done automatically because a data conversion is required.

Table 44 shows the type of validation done for every possible mapping between application data types (down) and EDI standard data types (across) during a TRANSLATE TO STANDARD processing.

Table 44. Type of validation during Translate to Standard

Data type:	A:	AN:	BIN:	N:	R:	DT:	TM:
A	A	R	AN	N	R	DT	TM
AN	A	R	AN	N	R	DT	TM

Table 44. Type of validation during Translate to Standard (continued)

Data type:	A:	AN:	BIN:	N:	R:	DT:	TM:
AC	A	R	AN	N	R	DT	TM
CH	CH	R	CH	N	R	DT	TM
DT	DT	DT	DT	DT	DT	DT	DT
TM	TM	TM	TM	TM	TM	TM	TM
R	R	R	R	R	R	DT	TM
N	N	N	N	N	N	DT	TM
P	—	—	—	—	—	DT	TM
L	—	—	—	—	—	DT	TM
Z	—	—	—	—	—	DT	TM
B	-	-	-	-	-	DT	TM
I	-	-	-	-	-	DT	TM
FN	CH	R	CH	CH	CH	CH	CH
H	-	-	-	-	-	DT	TM

Table 45 shows the type of validation done for each possible mapping between application data types (down) and EDI standard data types (across) during TRANSLATE TO APPLICATION processing.

Table 45. Type of validation during Translate to Application

Data type:	A:	AN:	N:	R:	DT:	TM:
A	A	A	N	R	DT	TM
AN	AN	AN	N	R	DT	TM
AC	AN	AN	N	R	DT	TM
CH	CH	CH	N	R	DT	TM
DT	DT	DT	DT	DT	DT	DT
TM	TM	TM	TM	TM	TM	TM
R	R	R	R	R	DT	TM
N	N	N	N	R	DT	TM
P	R	R	N	R	DT	TM
L	R	R	N	R	DT	TM
Z	R	R	N	R	DT	TM
B	R	R	N	R	DT	TM
I	R	R	N	R	DT	TM
FN	CH	CH	CH	CH	CH	CH
H	R	R	N	R	DT	TM

Appendix A. Mapping Binary Data

The ASC-X12 Specifications/Technical Information transaction set (841) defines a way for trading partners to exchange technical information the same way they exchange EDI transactions. This technical information, which can be graphic, image, or audio, can contain binary data. The binary data can assume any value in the range X'00' rough X'FF'.

In the syntax of X12 transaction sets, data elements that are separated by delimiters are combined into a segment that is identified by a segment ID and terminated with a segment delimiter. The binary data introduced by the 841 transaction set causes problems for this syntax because the binary data can contain a value that matches a segment delimiter. Translators and networks that support the 841 transaction set must have a way to identify binary data and determine its length so that it does not interfere with parsing the rest of the envelope. Special care must be taken if you want to send and receive files between and translators on other platforms. Not all operating systems support the record types uses. For more information, see “Format specifications” on page 289.

The BIN segment ID

The binary data is identified with a BIN segment ID, which notifies the parser that data following the segment ID is binary. Although the parser must always treat the BIN segment as if it contained binary data, the segment can contain normal text. The first data element of the BIN segment contains the length of the binary data so that the parser knows the amount of data to pass without interference. The first character after the binary data must be BIN segment terminator. Any other value is a syntax error that ends parsing for the envelope.

The BIN segment ID triggers the special binary processing. Although the 841 transaction set is the only one that uses the BIN segment, binary processing is not limited to the X12 standard. In addition, applies this special processing to all envelope types.

Length of the BIN segment

The value in the length data element of the BIN segment can be up to 15 characters long, which means the maximum length of a BIN segment is 999,999,999,999,999 bytes (fifteen 9s). However, the maximum size that supports is 2,147,483,647 bytes, which is the maximum signed integer value that a fullword can hold. The binary segment is a repeating segment with an unlimited number of repetitions, therefore it is unlimited for EDI.

Data Transformation for Binary Data

This section covers mapping binary data for Data Transformation to and from an application field.

BIN segment ID

Mapping a BIN segment

The BIN segment, as mentioned earlier, can have a rather impressive length. A new segment BDS has been introduced to transmit binary data in X12V4R4, transaction 102 (Associated Data).

The BIN and BDS segments

The binary data is identified with a BIN or BDS segment ID, which notifies the parser that data following the segment ID is binary. Although the parser must always treat the BIN and BDS segments as if it contained binary data, the segment can contain normal text.

The first data element of the BIN segment contains the length of the binary data so that the parser knows the amount of data to pass without interference.

The second data element of the BDS segment contains the length of the binary data so that the parser knows the amount of data to pass without interference.

The first character after the binary data must be BIN or BDS segment terminator. Any other value is a syntax error that ends parsing for the envelope.

The BIN and BDS segment IDs triggers the special binary processing.

Although using a file name is the primary way of providing data for a binary Segment, with Data Transformation maps you provide the data by passing it to WebSphere Data Interchange in application fields, the way all other application data is passed. The binary data itself must be defined in the Data Format with a data type of BN. The application can provide the length of the binary data. If the length of the binary data is not provided by the application the length will be the length of the application data field defining the binary data.

However, WebSphere Data Interchange has a maximum length of 32767 bytes for application fields, which is significantly less than the 999,999,999,999 bytes defined by the standard or the 2,147,483,547 bytes for WebSphere Data Interchange. WebSphere Data Interchange knows that it is dealing with binary data, and that some special processing is needed to put it into a BIN or BDS segment. The easiest way to pass this binary data to WebSphere Data Interchange is with a data format such as:

```
BINARY_LOOP1 999 Repeating loop
BINARY_STRUCTURE1 Record
    LENGTH N0 15 15-byte length field
    BINARY_DATA_LOOP 99 Repeating loop
        BINARY_DATA1 Record
            BINARY_FIELD BN 32767 Binary data field
```

Both BINARY_STRUCTURE1 and BINARY_DATA1 are passed separately records.

Now the application provides WebSphere Data Interchange with a BINARY_STRUCTURE1 and as many BINARY_DATA1 records as necessary to satisfy the length specified in the LENGTH field. WebSphere Data Interchange builds one BIN or BDS segment for each occurrence of BINARY_STRUCTURE1.

The BIN and BDS segment has a length value as specified by the LENGTH field and contains data from the BINARY_DATA1 records.

The EFI segment

For the 841 transaction set, an Electronic Format Identification (EFI) segment precedes a group of repeating BIN segments. The EFI segment provides information about the binary data file, provides special processing for only the following data elements in the EFI segment:

- File name
- Block type
- Record length
- Block length

Data elements such as *security technique* and *compression technique* do not receive special processing. When data compression is used, the sender must compress the data before translation, and the receiver must decompress it after translation. Any security that occurs is for the entire transaction or for the group containing the transaction. does not provide a special security interface for data in the BIN segments.

Use of the four data elements are covered later in the information pertaining to send and receive processing.

The following information can help explain how treats two of them: file name and block type.

File name: The EDI standard essentially defines file name as free format with a maximum length of 64 characters. Its format depends on the computer operating system being used. Accordingly, treats this element as a fully qualified data set name, including the owner ID. A file name value with a length greater than 44 characters (the maximum length of a data set name) is ignored.

Block type: The standard defines block type as free format with a maximum length of 4 characters. Its value indicates the organization of data in the BIN segments. Examples are fixed length, variable length, and spanned.

However, the definition does not provide any codes to represent the organizations, such as F for fixed and V for variable. In the absence of standard codes, interprets the block type as a string of characters that can have the following values:

A	ASA printer control characters
B	Block
F	Fixed
M	Machine code printer control characters
S	Spanned
U	Undefined

V Variable

Send processing for the binary segment

This section covers mapping a file or an application field to a binary segment.

Mapping data from a file to a binary segment

The primary intent of the 841 transaction set and the BIN segment is to transmit files between trading partners. Therefore, provides a way for you to map a *file* to a data element in the EDI standard. Normally, you would map a data element to an application field. Mapping data from a file to the BIN segment requires a new application data type, FN (file name).

The FN data type has special meaning only when it is mapped to the BIN02 data element of the BIN segment. At all other times, the FN data type is treated as an alphanumeric (AN) field. The special meaning for the FN data type when mapped to the BIN02 element is as follows:

- Rather than moving the field containing data to BIN02, moves the entire contents of a file named by the field to the BIN02 data element.
- automatically supplies the length (BIN01), based on the amount of data read from the file.

Aside from the special processing, the mapping of an FN field is the same as any other field, including the use of a translation table or user exit routine. For example, the application might not know the data set name of a file to be transmitted. The application might know it by a coded name. A translation table or a user exit provides a way of transforming the coded name to a data set name. You can also use a literal if the application data does not contain a value at all or if the value it supplies is all blanks. Because the name of the file mapped to the BIN segment can also be specified in the EFI segment, you must use the same field for mapping both segments.

If an EFI segment is built and it contains a value in the file name data element, that file name is used if the field mapped to the BIN02 data element is all blanks and no literal value is supplied.

The format of data in an FN type field is:

type:name

In this format, type is an optional value that indicates the type of file being provided. Valid values are as follows:

- DD** Data definition name (ddname)
- DS** Data set name
- MQ** queue profile member name
- VS** VSAM entry sequenced data set
- TD** Transient data queue

- TM** Temporary storage queue (main)
- TS** Temporary storage queue (auxiliary)

Note: VS, TD, TM, and TS are for CICS® only. The default is DS in and TS in CICS.

The name is either a ddname or data set name based on the value of type. It must conform to the conventions for its type. A ddname has a maximum length of 8. A data set name has a maximum length of 44.

The value of name can also be the literal &IV, which tells to substitute a value that represents the current binary file being processed. Each time a binary file is processed, increments this internal number. Thus, a specification of DD:EDIBF&IV equates to a ddname value of EDIBF1 the first time a binary file is processed, EDIBF2 the second time a binary file is processed, and so on.

The BIN segment, as mentioned earlier, can have a rather impressive length, and you might think that the entire file can be put into a single BIN segment. This is not always true. The number of BIN segments required to hold any file depends on the size of the file and the format of records in the file.

Format specifications

A file must be transmitted in a form that permits the receiver to recreate an exact copy of the file.

For based systems, a file consists of individual records and each record consists of some number of characters. Records can either have a fixed format where each record has the same number of characters, or a variable format where each record has a variable number of characters. For variable length records, the number of characters in the record is maintained in a record header which is the first 4 bytes of the record and has a format of LLXX, where LL is the number of characters in the record (including the length of the LLXX field) and XX is reserved.

Files on other systems (such as Windows® or) do not have record boundaries but are treated as a stream of characters with record boundaries (if any) established by the program that is reading the stream. provides 8 different ways to combine data from a file into BIN segments so that the receiver of the file can recreate an exact copy of the file being sent. The 8 different ways represent combinations of how records are put into binary segments (either combined or individually) and the format of the record header within the BIN segment. There are four possible header formats for records within the BIN segment:

1. No header - the data is put into the binary without a header.
2. format - each record in the binary segment is preceded by a record header of the form LLXX.
3. SHORT format - each record in the binary segment is preceded by a record header of the form LL.
4. LONG format - each record in the binary segment is preceded by a record header of the form LLLL.

Format specifications

WebSphere Data Interchange transmits all files with a fixed format by combining the records into a single binary segment without any record headers. Because the file has a fixed format, the record headers are not needed for the receiving side to reconstruct the file.

WebSphere Data Interchange transmits all files with a variable format by sending each record within the file as a single binary segment. Again record headers are not used because they are not necessary to be able to reconstruct the file.

If the defaults are not satisfactory, (for example, you want to send a file with variable length records as a stream of data in a single binary segment) then a special literal value can be supplied at mapping time that overrides the defaults. This literal is specified when mapping the binary element in the BIN segment (BIN02). You can either type it in the literal field on the mapping page, or map an application field to the BIN02, which at run time supplies the literal to . This literal consists of a series of keywords. Except for the first keyword (BinSpec), you can specify them in any order. In the following list, the keywords are shown with acceptable abbreviations in CAPS:

BinSpec

Signals that binary specifications follow.

BINary(C)

Indicates records are combined into a single binary segment (default for fixed record lengths).

BINary(R)

Indicates each record is a binary segment (default for variable record lengths).

Format(N)

No header precedes each record in the binary segment (default).

Format(V)

Use a header of the form LLXX, where LL is the number of bytes in the record and XX is binary zeros.

Format(S)

Use a header of the form LL, where LL is the number of bytes in the record.

Format(L)

Use a header of the form LLLL, where LLLL is the number of bytes in the record.

LIMIT(*nnnnn*)

If records are being combined, then *nnnnn* is the maximum size for a binary segment. After a binary segment reaches this limit, it is stopped and a new binary segment is started. The default limit is over 2 gigabytes or the amount of virtual storage available to the program, whichever is lower.

Examples

Sending PC executable files

In order to send a Windows or executable file received from a personal computer back to a personal computer, use the combined and no headers options:

BINSPEC BINARY(C) FORMAT(N)

Note: A variable length file sent with these options cannot always be recreated, because the records have been combined and no record length information has been added. This format is needed to send data to some PC translators.

Sending variable length files to PCs

In order to send a variable length file (such as a LIST3820) to a personal computer, so that none of the record length information is lost, even if all of the records are combined into one file, use the combined and any format option other than N.

BINSPEC BINARY(C) FORMAT(S)

In order to convert a file formatted as described back to its original format, use the same options on receive processing.

Sending a file to a system with a limit

Some systems have a limit to the size of a binary segment they can process. In order to limit the size of the data inside the BIN02 element to 1000 bytes, use the following option:

BINSPEC LIMIT(1000)

Note: A variable length file with records larger than 1000 bytes sent with these options cannot always be recreated, because the records have been split and no record length information has been added. The format parameter can also be used to add record length information.

Mapping an application field to a binary segment

Although using a file name is the primary way of providing data for a binary segment, you might find it useful to provide the data by passing it in application fields, the way all other application data is passed. If you do this, the application provides the length of the binary data and the binary data itself.

However, has a maximum length of 999 bytes for application fields, which is significantly less than the 999,999,999,999,999 bytes defined by the standard or the 2,147,483,547 bytes defined by WebSphere Data Interchange. knows that it is dealing with binary data, and that some special processing is needed to determine the length of data being passed, gather it, and put it into a BIN segment. The easiest way to pass this binary data to is with a data format such as this:

```
BINARY_LOOP1 999 Repeating loop
  BINARY_STRUCTURE1 Record
    LENGTH N0 15 15-byte length field
    BINARY_DATA_LOOP 99 Repeating loop
      BINARY_DATA1 Record
        BINARY_FIELD CH 999 Binary data field
```

Both BINARY_STRUCTURE1 and BINARY_DATA1 are passed separately. Now the application provides with a BINARY_STRUCTURE1 and as many BINARY_DATA1

Binary segment: mapping from application field

structures as necessary to satisfy the length specified in the LENGTH field. builds one BIN segment for each occurrence of BINARY_STRUCTURE1.

The BIN segment has a length value as specified by the LENGTH field and contains data from the BINARY_DATA1 structures.

If the BINARY_DATA1 structures you provide exceed the length you specified, ignores the extras. If you provide fewer than needed, binary zeros are added to the BIN segment until the LENGTH value is satisfied. The number of repetitions you specify for the structures is not important. accepts as many as the application passes.

As another example, suppose the application wants to pass in data from the file and have structures built in the same way described for variable-length files. The normal variable-length record format for records is a binary halfword containing the length of the data followed by the data itself. The data format definition required to accomplish this is:

```
BINARY_LOOP 999 Repeating loop
  BINARY_STRUCTURE2 Record
    LENGTH I0 2 2-byte binary length (the I0 is I (as in I am) 0)
  BINARY_DATA_LOOP 32767 Repeating loop
    BINARY_DATA2 ST Record
      BINARY_FIELD CH 1 Binary data field
```

BINARY_DATA2 is defined as physically part of its parent. Its maximum use count is 32756, because that is the maximum logical record length for physical sequential files.

The only difference between this example and the previous one is in the way data is passed to . In the first case, BINARY_STRUCTURE1 is passed followed by multiple BINARY_DATA1 structures. In the second case, only BINARY_STRUCTURE2 is passed because BINARY_DATA2 is defined as physically part of BINARY_STRUCTURE2.

There is a trade-off in choosing between the two cases. Structures always have a fixed length in . Therefore, when is presented with BINARY_STRUCTURE2, it always expects to get 32758 bytes. On the receive side, it always creates 32758 bytes. In effect, each variable-length record is expanded to its maximum length. The method used with BINARY_STRUCTURE1 conserves main storage at the expense of making the application break up the data into 999-byte chunks, in order for to put it back together.

You might ask, "But a structure can repeat only 32757 times, and a field can be only 999 bytes long, for a maximum record size of 32,724,243 bytes? What if I want to pass in more than that as a single structure?" assumes that if the field mapped to the binary data element of the BIN segment is the first field of a structure, then the length of the *structure*, not the length of the field, is used to move data. Furthermore, if the structure is defined to be physically part of its parent, and its parent is defined to be physically part of its parent, and it has no other fields, uses the length of the parent structure. For example:

Binary segment: mapping from application field

```
BINARY_LOOP 999 Repeating loop
  BINARY_STRUCTURE3 Record
    LENGTH N0 15 15-byte length field
    BINARY_DATA_LOOP1 10000 Repeating loop
      BINARY_DATA3 Record structure
        BINARY_DATA_LOOPA 1000 Repeating loop
          BINARY_DATA3A Record
            BINARY_DATA_LOOPB 10000 Repeating loop
              BINARY_DATA3B Record
                BINARY_FIELD1 CH 999 Binary data field
                BINARY_FIELD2 CH 1 Binary data field
```

BINARY_DATA3, BINARY_DATA3A, and BINARY_DATA3B are all defined to be physically part of their parent. Using this definition and mapping the binary data field in the standard to BINARY_FIELD1, the translator arrives at a length value of 1000 (the length of BINARY_FIELD1+BINARY_FIELD2) * 10000 (the number of times BINARY_DATA3B repeats) * 10000 (the number of times BINARY_DATA3A repeats) * 10000 (the number of times BINARY_DATA3 repeats). This yields an effective length of 10 to the 15th (1 greater than defined by the standard).

Of course this structure cannot really be used because the maximum length defined by WebSphere Data Interchange is 2,147,483,647. Nevertheless, the example illustrates the technique of creating an effective length greater than 999.

Receive processing for the binary segment

This section covers mapping a binary segment to a file or an application field.

Mapping data from a binary segment to a file

Receiving presents a set of problems that are not present on the send side. During send processing, it is reasonable to assume that senders know what they are sending, when they are sending it, and the data set names of the files they are sending. Receivers, on the other hand, can have little control over what they receive, when they receive it, and how many files they receive at any one time. Therefore, must be capable of creating files when necessary.

As with sending, if you want the binary data passed to you in a file rather than the application fields, map the binary data to a field in the application defined with the FN data type. You do not receive the binary data in the field; instead you receive the name of the data set where the data was written.

On the send side, you can provide the file name in an application field buffer or as a literal value. You can also provide a literal on the receive side. If you provide a literal for an FN data type that is mapped to the binary data element of the BIN segment (BIN02), gives the literal special processing. Normally, a literal mapped for receiving is used only if the standard data element is blank. The special processing for this literal is that you can use it to provide the file name and allocation parameters that needs to create the file on your system.

The literal has this format:

Binary segment: mapping to file

type:name parameters

type

An optional value that indicates the type of file being provided. Valid values are:

DD	Data definition name (ddname)
DS	Data set name
MQ	queue profile member name
VS	VSAM entry sequenced data set
TD	Transient data queue
TM	Temporary storage queue (main)
TS	Temporary storage queue (auxiliary)

VS, TD, TM, and TS are for CICS only. The default is DS in and TS in CICS.

name

is either a ddname or data set name based on the value of *type*. It must conform to the conventions for its type. A ddname has a maximum length of 8 characters. A data set name has a maximum length of 44 characters.

name can also contain one of the following special literals to substitute other values in this field:

&IV	A value that represents the current binary file that is being processed. An F is inserted in front of this value if the IV value begins another level of qualification on the data set name.
&U	The current user ID. A dot (.) is inserted after the user ID to force another level of qualification on the data set name.
&D	The current date as a yymmdd value. D is inserted in front of this value if it begins another level of qualification on the data set name.
&T	The current time as an hhmmss value. A T is inserted in front of this value if it begins another level of qualification on the data set name.
&E	The current file name from the EFI segment. Another level of qualification is forced both before and after the EFI file name.

name is optional. If you do not provide it, the file name from the EFI segment is used. If an EFI segment does not exist or does not contain a file name field, a default data set name &U.BIN&V.D&D.T&T is used. The name that is used is returned to the application in the FN field. The maximum length of a data set name is 44 characters, and truncation occurs if necessary.

parameters

A series of keywords and values that supply the data needed to create a new data set on your system. Except for the first keyword (ALL), you can specify them in any order. The keywords are shown with an acceptable abbreviation in CAPS:

ALLocate

Signals that allocation parameters follow.

TRacKs

Allocation unit is tracks.

CYLinders

Allocation unit is cylinders. This is the default value if TRK or BLK is not specified.

BLocKs(*b*)

Allocation unit is BLOCKS with an average block length of *b*.

Space(*p*, *s*)

Primary space quantity of *p* (default value of 10). Secondary quantity of *s* (default value of 10).

UNit(*vvvv*)

vvvv is the unit name for the allocation. Default value is SYSDA.

UCount(*n*)

n is the number of units for the allocation. Default value is 1.

Lrecl(*l*) *l* is the logical record length. Default value is 32756 or the value taken from the EFI segment.

Blksize(*b*)

b is the block size. Default value is 32760 or the value taken from the EFI segment.

Recfm(*xxxx*)

xxxx is the record format for records in the file. Default value is VB or the value taken from EFI segment. Valid values are:

- A** ASA printer control characters
- B** Block
- F** Fixed
- M** Machine code printer control character
- S** Spanned
- U** Undefined
- V** Variable

LIKE(*dsname*)

dsname is the name of some other data set from which the LRECL, BLKSIZE, and RECFM are copied.

first tries to open a given data set name. If the attempt to open the data set is successful, the file is used, and any data in the file is overlaid by the new data. If the attempt fails, tries to create a new file using the allocation value (or default) shown. The new data set has a specification of (NEW,CATLG,CATLG), and all unused space in the data set is released when the data set is closed. For CICS, if the attempt to open the data set fails, a unique temporary storage queue is created, and the data is written to

Binary segment: mapping to file

this queue. Default values are overridden by values from the EFI segment, which in turn are overridden by literal values supplied at mapping time.

If the data being received is being written to a fixed-length file, assumes the binary segment consists of multiple records without any record headers. If the data being received is being written to a variable-length file, assumes each binary segment contains data for a single record without any record headers.

takes data from the binary segments and adds them to the file. For each BIN segment, the amount of data defined by the logical record length of the file is extracted and written to the file. This continues until the amount of data left in the BIN segment is less than or equal to the logical record length. If the amount of data left in a binary segment is less than the logical record length, fixed-length records are padded with binary zeros; variable-length records are written with the shortened logical record length. Data from the end of one BIN segment is never combined with the data from the beginning of the next BIN segment. At a minimum, each BIN segment defines the beginning of a new record in the file.

If the defaults previously described are not required, then the binary specifications described in “Format specifications” on page 289 (BINary and Format) can be provided as part of the allocation parameters in the receive literal value.

Mapping data from a binary segment to an application field

The considerations described for mapping from an application field to a binary segment also apply when mapping data from a binary segment to an application field. See “Mapping an application field to a binary segment” on page 291.

Appendix B. Hierarchical loops

A hierarchical loop (HL) is similar to an organization chart. Just as an organization chart shows you the various groups of people and their relationship to the whole, an HL shows you each group of data and its relationship to the whole. Figure 14 shows the different levels in the organization, and who reports to whom.

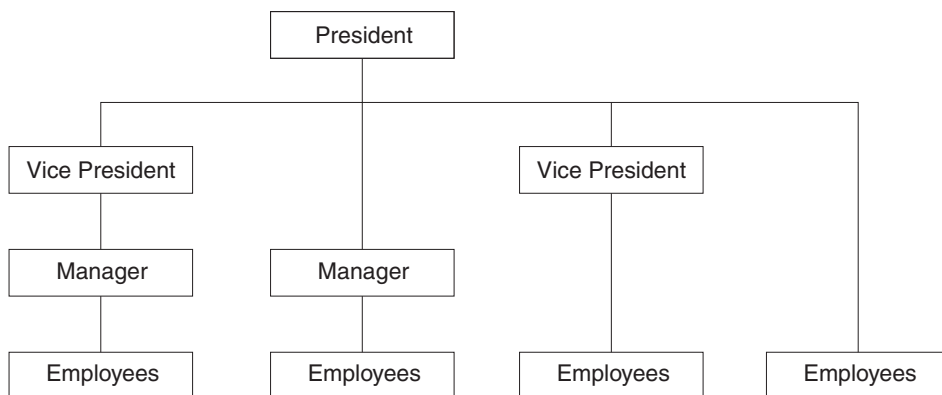


Figure 14. HL example 1

Hierarchical loops define different levels of data, which can be used in any sequence, and skipped when appropriate, enabling you to fit the loop to your data. See Figure 15.

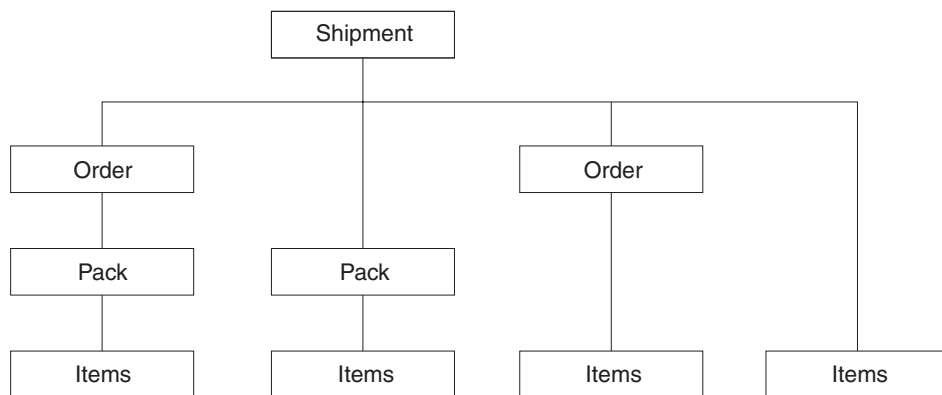


Figure 15. HL example 2

Specifying HL levels

Hierarchical loops are a way for EDI transactions to dynamically define the looping structure. The standard defines all segments which might be a part of a HL but it is the application/standard data being received which adds structure to those segments. For example an HL loop would be defined to contain segments A, B, C, D, E; but it is the

Hierarchical loops

application/standard data which defines that segments A,B,C form one level 1 inner loop, segments D,E form another level 1 inner loop and further that segments D,E might form a level 2 inner loop within the A,B,C loop. Theoretically there is no limit to the nesting levels.

HL loops can be qualified and mapped using the usual qualification and mapping methods described in this book. However, with the HL support in WebSphere Data Interchange, you can define Hierarchical levels and specify unique mapping instructions for each identifiable group of structures in a hierarchical level (loop).

Note: It is not required to use special HL Qualification when specifying an HL.

The HL segment

Sometimes you might need to nest loops several levels deep to map all of your data, yet each level contains similar information, such as name and address. In the following figure, loops are nested four levels deep, and the N1 and N3 segments occur at the beginning of each loop.

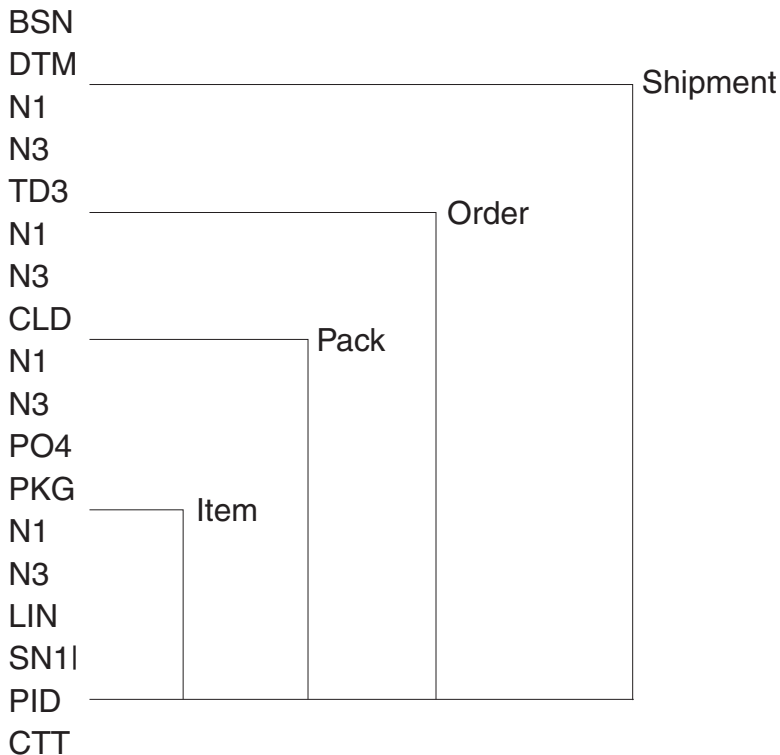


Figure 16. The HL segment

The hierarchical level (HL) segment makes it easier to map these loops. Each HL segment contains information about the relationship of segments in a hierarchical loop

to the other segments in the loop. This information is described in Table 46.

Table 46. The HL segment

Field ID	Field Name	Description
HL01	ID number	A unique number that identifies the occurrence of the HL segment. This data element is alphanumeric and has a maximum length of 12 characters. This field usually contains a sequential number that increments for each occurrence of the HL segment.
HL02	Parent ID	The HL01 value of the HL segment that is the parent of the current HL segment.
HL03	Level code	A code that indicates the level of the HL segment in the current HL loop. For example, the level code can refer to the shipment, order, or item level information in the ANSI X12 Shipping Notice transaction set.
HL04	Child code	A code that indicates if the segment has subordinate segments: 1 for subordinate segments, or 0 for no subordinate segments. The default is 0.

HL01 and HL02 provide the information for WebSphere Data Interchange to determine the nesting of loops within each other.

Note: The HL segment is not supported by all standards.

Preparing hierarchical loops

HL support enables you to specify unique mapping instructions for each identifiable group of structures in a hierarchical loop. WebSphere Data Interchange can handle 16 levels of nesting within the HL loop structure. To begin mapping a hierarchical level loop, follow these steps:

1. Create the hierarchy for your application data.
2. Assign node IDs to your hierarchy to uniquely identify logically grouped segments in the hierarchical loop. This ensures that WebSphere Data Interchange processes your data the way you want it to be processed. The logical grouping of segments within the HL and the hierarchical relationship is defined by standards organizations or industry groups. Trading partners then agree which segments they will use and how they will be mapped.

Mapping service segment fields

3. Create the application data format for this hierarchical loop. The data format and the HL mapping tell WebSphere Data Interchange how to build the hierarchical loop.

Figure 17 shows how each group of segments in your hierarchy is numbered in a top-down, left-right order. Use this number as the node ID value.

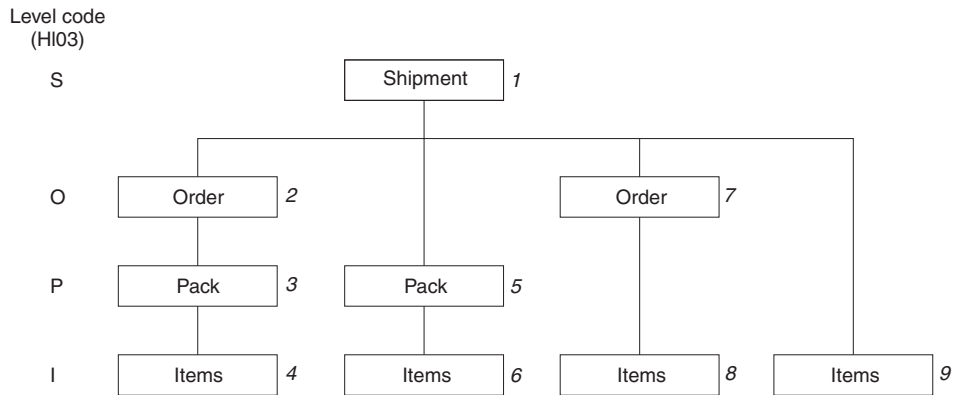


Figure 17. Preparing Hierarchical Loops

Data Transformation Mapping for HL Loops

WebSphere Data Interchange provides special mapping support for the HL loop which defines Hierarchical levels and specifies unique mapping instructions for each identifiable group of structures in a hierarchical level (loop). For EDI target messages the HL segment can be automatically created eliminating the need for the application to supply the hierarchical information.

Creating a Data Transformation Map

Source based mapped is required to use special HL mapping support when EDI is the source message.

Target based mapped is required to use special HL mapping support when EDI is the target message.

For information about creating a Data Transformation map, see Chapter 9, “Data Transformation mapping,” on page 121.

Defining HL loop levels

Defining HL loop levels. To create the base HL level:

1. Go to the mapping details tab, and right-click the HL loop in the command window. The Qualification Selection Window opens.
2. Select Add HL Qualification.
3. Select an HL level code from the list. The list contains valid codes from the code list for HL03 (element 735).

An HLLLevel command is created in the command window as follows:

```
HLLLevel(char "Level code")  
HLLLevel('S')
```

To create the underlying HL nesting levels or children for this HL level, right-click the HLLLevel command and select HL Qualification. The AddChild HL Qualification and AddPeer HL Qualification commands display.

- To create a child node, select the AddChild command.

After selection of AddChild, select an HL level code from the list. The list contains valid codes from the code list for HL03 (element 735). The HL loop is inserted as a child of the current HL loop with an HLLLevel command. The HL segment is automatically selected and all elements mapped with an HLAutoMapped command. This command cannot be removed or modified. Additional mapping for these elements can be accomplished using normal mapping methods.

- To create a sibling node, select the AddPeer command.

After selection of AddPeer, select an HL level code from the list. The list contains valid codes from the code list for HL03 (element 735). The HL loop is inserted as a sibling of the current HL loop with an HLLLevel command. The HL segment is automatically selected and all elements mapped with an HLAutoMapped command. This command cannot be removed or modified. Additional mapping for these elements can be accomplished using normal mapping methods.

With each HL loop level created, the HL loop is copied to its hierarchical location (as a child or peer) with an HLLLevel command, and displayed in the command window as if the hierarchy were explicitly defined in the standard.

Qualifying HL loop levels in an EDI source message

To qualify the HL level, right click the HLLLevel command and select the Qualify command to select qualification by occurrence, value, or expression. For multiple occurrence qualification, drag the source path from the source window to the target path in the target window or drag the target path from the target window to the HL loop in the mapping command window. This creates a MapTo() command under the HLLLevel command in the command window.

Qualifying HL loop levels in an EDI target message

This creates a ForEach() command under the HLLLevel command in the command window. For additional qualification using occurrence, value, or expression, right click the ForEach command and select the Qualify command.

If you do not want a ForEach command (for multiple occurrence qualification), occurrence is assumed.

To add qualifications using value or expression, right click the HLLLevel command, select Insert Within, and use conditional commands.

For information about commands not described in this section, see Chapter 12, "Data Transformation mapping commands and functions," on page 141.

Mapping service segment fields

Handling special HL mapping for Data Transformation maps

In some cases, you might find it necessary to create a special HL mappings for a Data Transformation map. A special HL mapping does not use Parent IDs in input data.

Note: Special HL mapping is used only if the source data does not contain the values for an HL segment. It is not required to use special HL Qulaification when mapping an HL loop.

Source based HL mapping

Source based mapping is required to use special HL mapping support when EDI is the source message.

If special HL mapping is used, the EDI parser constructs the HL Hierarchy in the abstract message. HL Loops are then placed as a child of their parent.

Then the transformation executes the mapping instructions based on the hierarchy in the map and the abstract message. Otherwise the abstract message is re-constructed as a flat hierarchy. If your source message does not contain the parent and child elements in the HL segment, the hierarchy will be flat and the HL loops are all siblings.

Figure 18 is an example of an EDI that has no hierarchy.

```
ST*856*45920001~  
BSN*30*01*050524*0049~  
HL*1**S~  
N1*SF**92*SHIP1~  
HL*2**0~  
PRF*ORD1*11113711**050523~  
HL*3**P~  
MAN*GM*PAC1111~  
HL*4**0~  
PRF*ORD2*11113712**050523~  
HL*5**P~  
MAN*GM*PAC2222~  
HL*6**S~  
N1*SF**92*SHIP2~  
HL*7**0~  
PRF*ORD3*11113711**050523~  
HL*8**P~  
MAN*GM*PAC3333~  
HL*9**0~  
PRF*ORD4*11113712**050523~  
HL*10**P~  
MAN*GM*PAC4444~  
CTT*10~
```

Figure 18. Example of EDI no hierarchy (no parent id HL02)

Target based HL mapping

Target based mapping is required to use special HL mapping support when EDI is the target message. If special HL mapping is used, the transformation creates the hierarchy with the HL level code value HL03 in the abstract message and HL Loops are placed as a child of their parent. The EDI serialization generates HL01, HL02, and HL04 values based on the parent/child relationship in the abstract message produced by the transformation.

With XML source, the DTD does not always have the parent/child relationship for the hierarchy and there is no proper hierarchical nesting for mapping. Although the XML source input data can look like it contains the hierarchy with spacing and indentation, it is free form. The only way to identify the hierarchy is by using the DTD or Schema definition.

Figure 19 shows an XML DTD with nesting and Figure 20 on page 304 show and XML DTD with no nesting.

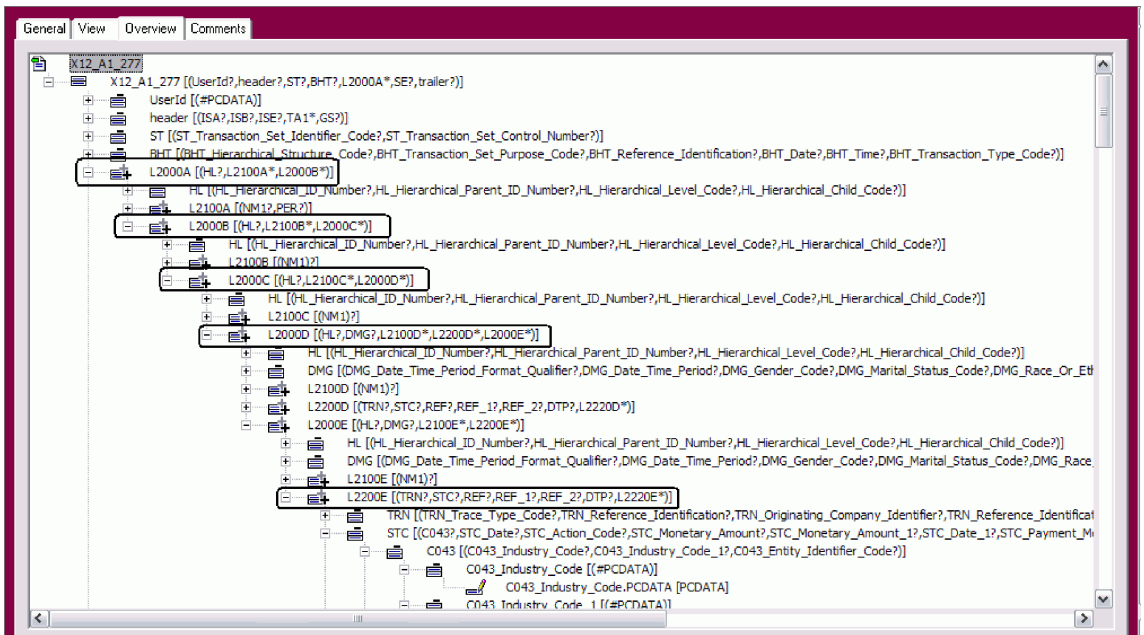


Figure 19. XML DTD with nesting

Mapping service segment fields

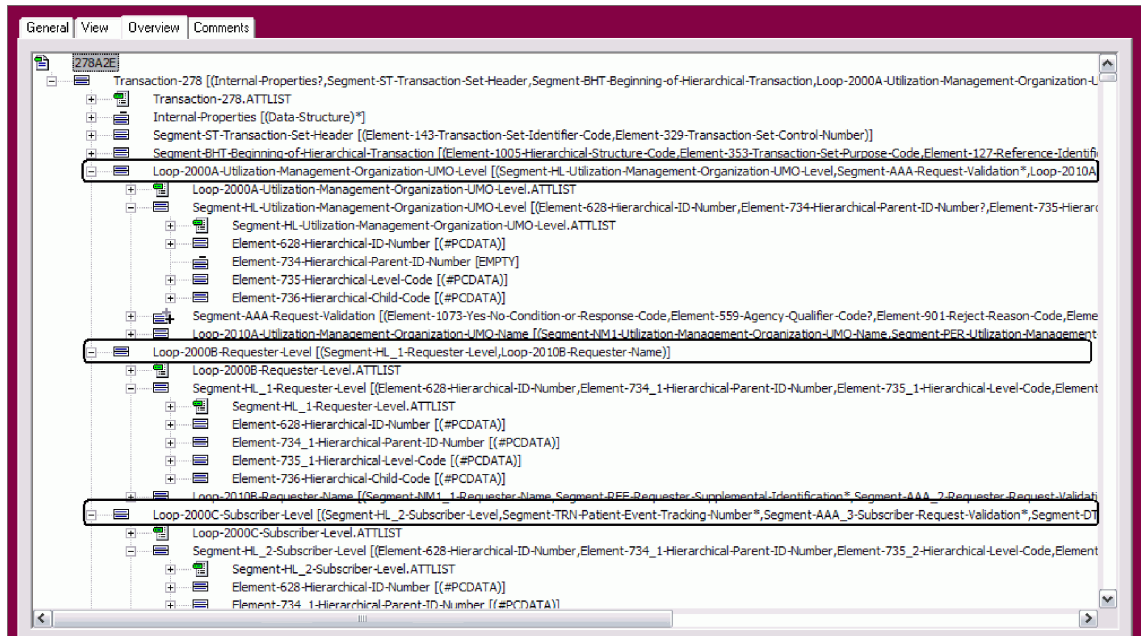


Figure 20. XML DTD without nesting

Creating a special HL qualification

The special HL Loop mapping is accomplished by using the HL Loop Level codes. Drag-and-drop operations cannot be used.

Creating the base HL level:

1. Go to the mapping details tab, and right-click the HL loop in the command window. The HL Qualification window opens.
2. Select Insert HL Qualification.

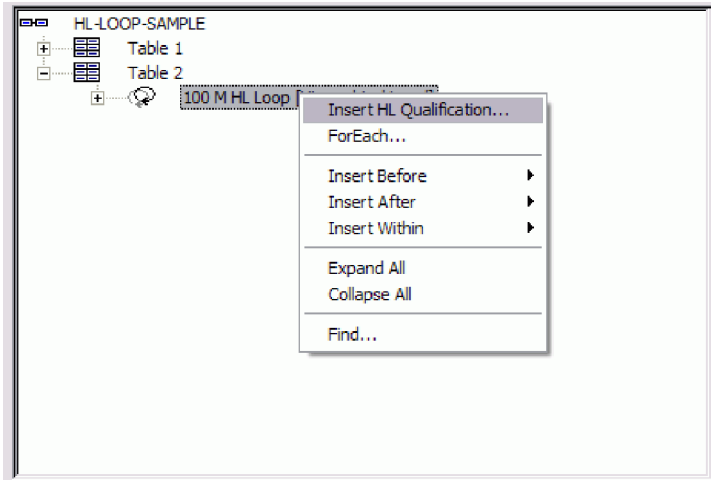


Figure 21. Loop menu

3. Select an HL level code from the list. The list contains valid codes from the code list for HL03 (element 735).

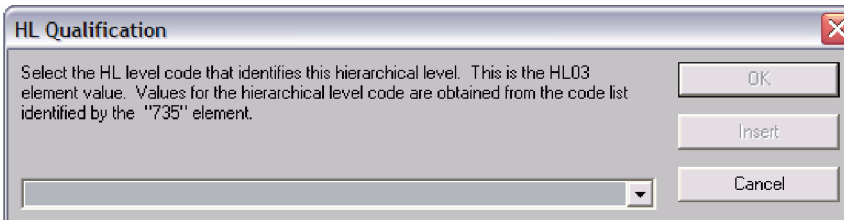


Figure 22. HL Qualification window

4. Click OK. An **HLLevel** mapping command is created in the command window as follows:
 - HLLevel (char "level code")
 - HLLevel ('1')

Mapping service segment fields

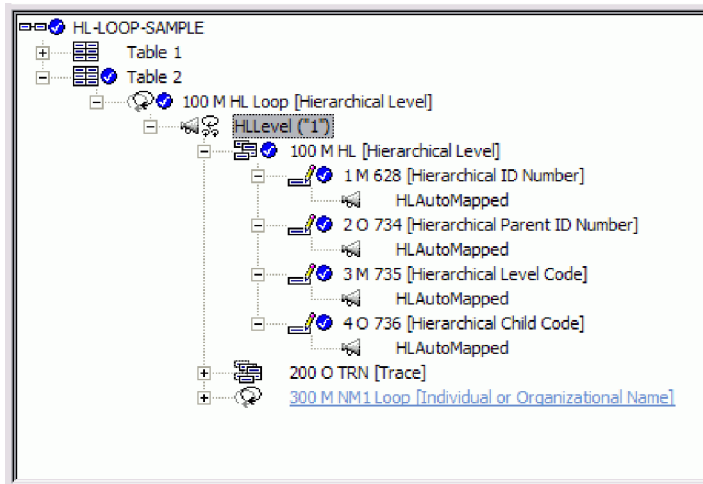


Figure 23. HLLevel command

The HL segment is automatically selected and all elements mapped with an **HLAutoMapped** command. This command cannot be removed or modified. Additional mapping for these elements can be accomplished using normal mapping methods.

Creating a default qualification

With each HL loop level created, the HL loop is copied to its hierarchical location (as a child or peer) with an **HLLevel** command and displayed in the command window as if the hierarchy were explicitly defined in the standard. To create a default HL Level mapping as a base level, right-click the HL loop in the command window and select **Insert HL Default Qualification**. This selection is only available with EDI Source based maps. The HL loop is inserted as a sibling of the current HL loop with an **HLLevel** command.

Adding levels to an HL qualification

To create the underlying HL nesting levels or children and peers (siblings) for a HL level:

Right-click a **HLLevel** command and select **HL Qualification**. The **Insert Child HL Qualification** and **Insert Peer HL Qualification** commands display.

- To create a child node, select the **Insert Child HL Qualification** command.

Select an HL level code from the list. The list contains valid codes from the code list for HL03 (element 735). The HL loop is inserted as a child of the current HL loop with an **HLLevel** command.

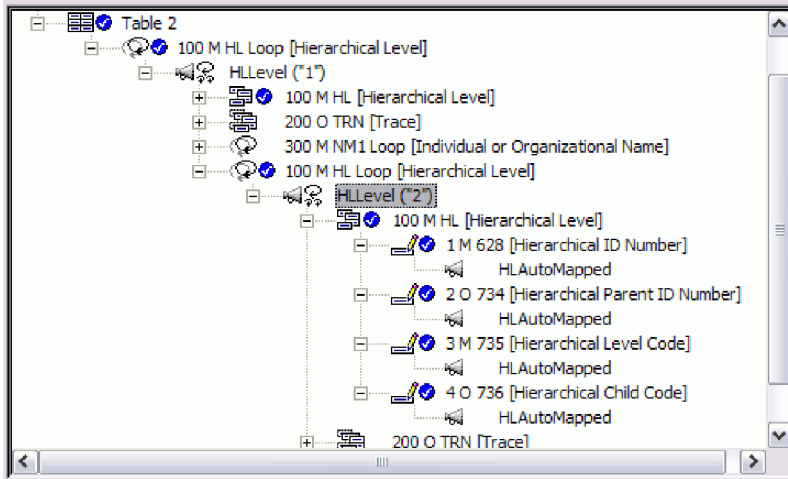


Figure 24. A child HL qualification

- To create a peer (sibling) qualification, select the **Insert Peer** command. Select an HL level code from the list. The list contains valid codes from the code list for HL03 (element 735). The HL loop is inserted as a sibling of the current HL loop with an **HLevel** command.

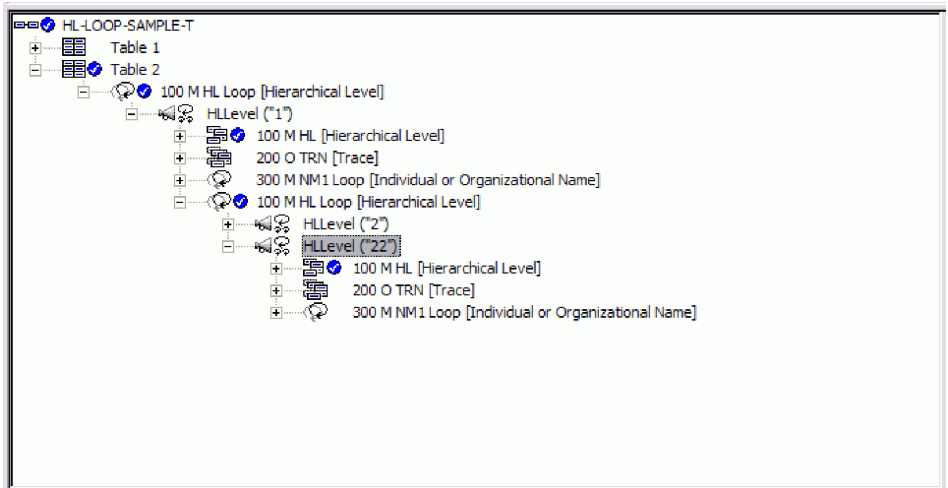


Figure 25. A peer (sibling) HL qualification

Comparing HLevel and Qualify by value

- The HLevel mapping command behavior is similar to using a Qualify by value Qualification for EDI source maps.
 - Qualify (StrComp (path, "value") EQ 0)

Mapping service segment fields

- HLLevel ("1")
- The difference between the Qualify by value and the HLLevel are:
 - There is a path associated with Qualify by value.
 - There is no path associated with HLLevel.
- Additional Qualification, such as a multi-occurrence qualification **MapTo** and **ForEach** commands, can be placed under the **HLLevel** mapping command by using the drag-and-drop technique. Other qualification can be added with a right click **HLLevel** mapping command.
- Adding additional qualification to the **HLLevel** mapping command activates the implied **CloseOccurrence**.
- If no additional qualification is added, a **CloseOccurrence** command might be needed.

Example of no qualification under HLLEVEL

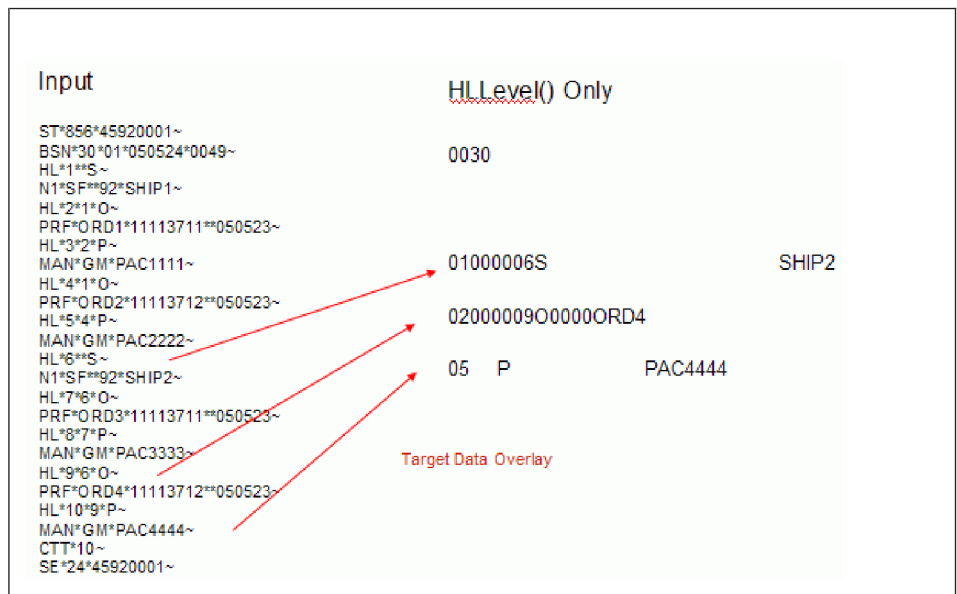


Figure 26. No qualification under HLLEVEL data example

Using CloseOccurrence with no qualification

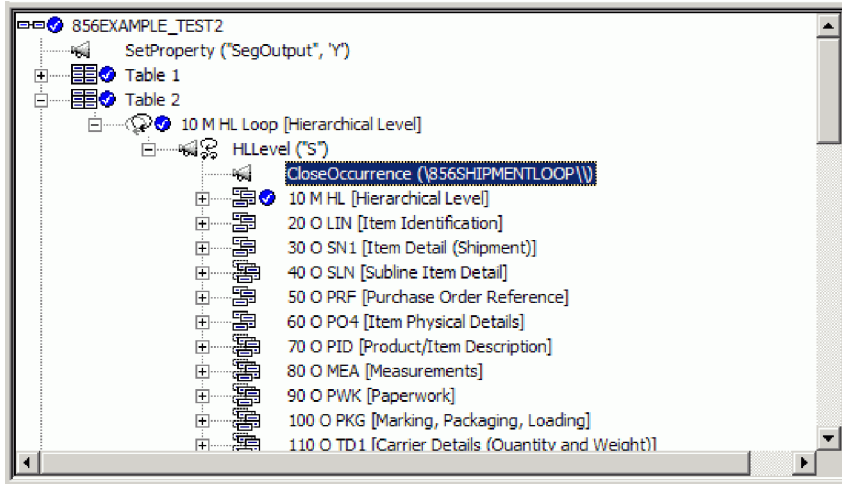


Figure 27. CloseOccurrence command with no qualification example

Input	HLLevel() Only: CloseOccurrence 'S' Level	
ST*856*45920001~		
BSN*30*01*050524*0049~	0030	
HL*1**S~		
N1*S F**92*SHIP1~		
HL*2*1*O~	01000001S	SHIP1
PRF*ORD1*11113711**050523~		
HL*3*2*P~		
MAN*GM*PAC1111~	02000004O0000ORD2	
HL*4*1*O~		
PRF*ORD2*11113712**050523~		
HL*5*4*P~	05 P	PAC2222
MAN*GM*PAC2222~		
HL*6**S~		
N1*S F**92*SHIP2~		
HL*7*6*O~	01000006S	SHIP2
PRF*ORD3*11113711**050523~		
HL*8*7*P~		
MAN*GM*PAC3333~		
HL*9*6*O~	02000009O0000ORD4	
PRF*ORD4*11113712**050523~		
HL*10*9*P~		
MAN*GM*PAC4444~		
CTT*10~	05 P	PAC4444
SE*24*45920001~		

Overlay on Order

Figure 28. CloseOccurrence command with no qualification incorrect data example

Mapping service segment fields

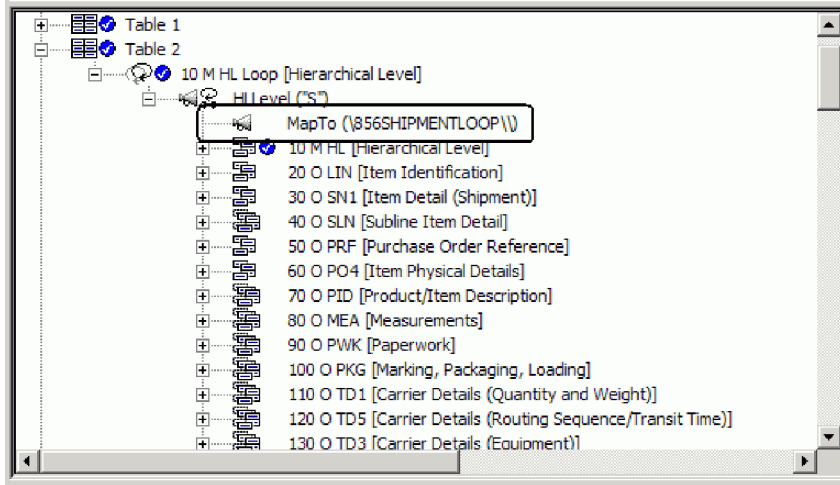


Figure 29. Multi-occurrence qualification example

Input	Output	
ST*856*45920001~	0030	
BSN*30*01*050524*0049~		
HL*1**S~	01000001S	SHIP1
N1*SF**92*SHIP1~		
HL*2*1*O~	02000002O0000ORD1	
PRF*ORD1*11113711**050523~		
HL*3*2*P~	05 P	PAC1111
MAN*GM*PAC1111~	02000004O0000ORD2	
HL*4*1*O~	05 P	PAC2222
PRF*ORD2*11113712**050523~		
HL*5*4*P~	01000006S	SHIP2
MAN*GM*PAC2222~		
HL*6**S~	02000007O0000ORD3	
N1*SF**92*SHIP2~		
HL*7*6*O~	05 P	PAC3333
PRF*ORD3*11113711**050523~	02000009O0000ORD4	
HL*8*7*P~	05 P	PAC4444
MAN*GM*PAC3333~		
HL*9*6*O~		
PRF*ORD4*11113712**050523~		
HL*10*9*P~		
MAN*GM*PAC4444~		
CTT*10~		
SE*24*45920001~		

Figure 30. Multi-occurrence qualification data example

Mapping the HL segment in a send or Receive map

WebSphere Data Interchange provides special handling for Hierarchical Loops. This section shows how to map the HL segment using the WebSphere Data Interchange Client interface.

1. Double-click an HL Loop.

- The Hierarchical Loop Support window opens.
2. Click Special HL Support.
The Qualify a Hierarchical Loop window opens.
3. Type the ID of the node you are mapping in the **Node Number** field. Click the question mark icon for field descriptions.
4. Select a hierarchical level code from the Hierarchical Level Code (HL03) list. Click the question mark icon for field descriptions.
5. If you need to repeat the segment mapping, click Repeat.
The Hierarchical Loop Support window reopens.
6. When you have completed all repeat mappings, click OK.
The words Qualified by HL Logic. . . display next to the segment name in the Mapping editor.

HL segment literal keywords for Send and Receive Maps

Your application data cannot contain the information the translator needs to create the HL segment, but you can use the following special literals to supply the values for the HL segment with a Send or Receive Map.

Table 47. HL segment literal keywords

Keyword	Description
&HLID	Supplies a sequential number for each HL segment created.
&HLPID	Supplies the HLID value for the parent of the current HL.
&HCODE	Supplies the hierarchical code associated with the current HL segment.
&HCHILD	Supplies the value 1 if the current HL segment has subordinate segments.

When you map the HL segment, you can use these literal keywords by typing them in the Literal field of the Map Data Element panel (TP10) when you map the data elements. For send, they will be automatically mapped with their corresponding special literal when the HL loop is qualified. You must remap these elements if you want different mappings.

- HL element 628 will be mapped with &HLID
- HL element 734 will be mapped with &HLPID
- HL element 735 will be mapped with &HCODE
- HL element 736 will be mapped with &HCHILD

For more information about HLs with send and Receive maps, see the WebSphere Data Interchange client help.

Appendix C. Handling international characters

WebSphere Data Interchange Version 3.3 now supports the translation of international data using Data Transformation maps and processing using code pages. The WebSphere Data Interchange database now supports Unicode characters. Unicode is a character set standard that is designed to include characters that appear in most languages. Currently, the Unicode standard contains 34,168 distinct coded characters derived from 24 supported language scripts. These characters cover the principal written languages of the world.

Unicode defines two mapping methods:

- Unicode Transformation Format (UTF)
- Universal Character Set (UCS)

WebSphere Data Interchange primarily uses the UTF-8 encoding which is an 8-bit, variable-width encoding, compatible with ASCII.

The addition of Unicode to WebSphere Data Interchange adds the following support:

- The use of any international character into WebSphere Data Interchange objects such as profiles, maps, and translation and validation tables. This removes the current restrictions on characters outside the default code page appearing in the database.
- The ability to enter international characters using the WebSphere Data Interchange Client.
- Enables the import and export of Unicode data, including international characters.
- The ability to specify the source and target encoding.
- The support of International characters in values such as EDI segment IDs.
- Multiple XML documents in an input file, even if the input encoding is not compatible with the local code page.
- XML tag names to contain characters that are outside the local code page.
- Remove limitation on current Data Transformation processing.

The following restrictions apply for international characters in WebSphere Data Interchange:

- The WebSphere Data Interchange Client interface, server error messages, and reports are presented in English only. However, the WebSphere Data Interchange Client opens and accepts international characters in the data fields. Server error messages and reports can, in some cases, display substitution characters from embedded data that contain international characters.
- The WebSphere Data Interchange batch utility will not process Unicode PERFORM commands. Certain names such as profile names and map names are commonly used in batch utility commands and in reports. Characters outside the local code page will not be supported for these fields.
- Send and Receive maps do not support Unicode and international characters.

Handling international characters

- The DTD Convert utility continues to support the local code page only. If user data containing international characters appears as substitution data in the messages, the international characters might be replaced by substitution characters.
- CICS transactions are used to start WebSphere Data Interchange and for specific WebSphere Data Interchange components or processing. These transactions are limited and do not handle international characters as parameters or inputs.

The following sections contain information about the enhancements in WebSphere Data Interchange that enable the handling of International characters.

Note: Use of international characters in object names is not restricted, but not recommended. Using characters that cannot be represented in the local code page for the object names can prevent them from being used or displayed properly when used or referenced in the **PERFORM** command or send and Receive maps.

Note: International characters are not supported in send and Receive maps. If used, the characters are either replaced with a substitution character or can result in a WebSphere Data Interchange error.

Processing and automatic detection for UNICODE

The SourceEncode and TargetEncode PERFORM keywords override the automatic detection process. The following section describe the processing and automatic detection of data.

Encoding used for incoming Application data

The encoding used to interpret incoming Application data is determined as follows:

- The data format definition code page field is used to determine the source application data encoding for raw data and C and D record formats.
- For C and D record formats, all UNICODE values and the LOCALCP (default system code page) are used to identify the C record.
- If no encoding is found, the LOCALCP value is used.

Encoding used for outgoing Application data

The encoding used to interpret outgoing Application data is determined as follows:

- The **SetProperty** mapping command using the EncodeTarget keyword determines the target application data encoding for raw data and C and D record formats
- The data format definition code page field is used to determine the target application data encoding for raw data and C and D record data.
- If no encoding is found, the LOCALCP (default system code page) value is used.

Encoding used for incoming EDI data

The encoding used to interpret incoming EDI data is determined as follows:

- For EDIFACT data only:

The EDI parser looks up the UNB0101 value in the EDISYNTAX table. This new translation table is supplied in the form of an EIF file to convert the EDIFACT syntax id (UNB0101) subelement to an encoding value.

For example, it converts UNOC to ISO8859-1, UNOD to ISO8859-2, and so on. Users can modify this table if needed, or omit it entirely if they do not want to use the EDIFACT encoding detection logic.

- If no encoding is found, the value LOCALCP (default system code page) is used.

Encoding used for outgoing EDI data

The encoding used to create outgoing EDI data will be determined as follows:

If no encoding was found, the EncodeTarget property set in the map is used. The availability of TP User fields as mapping properties enables you to specify different EncodeTarget values for different trading partners. For example, they can set the User Field 1 in the trading partner to a specific encoding, and include a mapping command such as the following:

```
SetProperty("EncodeTarget", GetProperty("ReceiverTPUser1"))
```

The following mapping properties from the sender and receiver trading partner profiles can be retrieved using the GetProperty mapping function in a map:

Property name

Property Description

SenderTPProfile

Sender trading partner profile name. If no profile is found for the sending trading partner, the value UNKNOWN is returned.

ReceiverTPProfile

Receiver trading partner profile name. If no profile is found for the sending trading partner, the value UNKNOWN is returned.

SenderTPUser1–SenderTPUser10

User fields from the sender's TP profile. If no sender TP profile was found, the user fields from the ANY trading partner will be used.

ReceiverTPUser1–ReceiverTPUser10

User fields from the receiver's TP profile. If no receiver TP profile was found, the user fields from the ANY trading partner will be used.

If no encoding is found, the LOCALCP (default system code page) value is used.

Encoding used for incoming XML data

The encoding used to interpret incoming XML data is determined as follows:

The encoding is determined by the normal XML auto-detection logic defined in the W3C XML recommendation, Appendix F.

Encoding used for outgoing XML data

The encoding used to interpret outgoing XML data is determined as follows:

- The **SetProperty** mapping command with the EncodeTarget keyword is used to determine the target XML data encoding.

Handling international characters

- If no encoding was found, the LOCALCP (default system code page) value is used.

Import and export considerations

Import accepts data in either UTF-8 format or in the local code page format. This enables the support of older EIF files and user-generated EIF files.

Export uses UTF-8 format to export data. Using UTF-8 exclusively with exports ensures that any international characters are preserved.

MQ profile fields

As result of UTF-8 implementation, two new MQ profile fields are available:

- **CNVTFIELD**
The CNVTFLAG value controls whether WebSphere Data Interchange uses get (without convert) or get-with-convert. The default is get-with-convert.
- **CCSID**
The CCSID value specifies the CodedCharSetId used on get-with-convert. This value is used only if the first property indicates that conversion is done. The default is to use the CCSID of the queue manager.

PERFORM keywords for international data

New PERFORM keywords associated with the Unicode enhancement are available for the **PERFORM TRANSFORM** command:

- SOURCEENCODING
- ENCODETARGET
- IGNOREBOM

SOURCEENCODING and ENCODETARGET

Any supported encoding value can be specified. The SOURCEENCODING and ENCODETARGET keywords are used with the PERFORM TRANSFORM command and provide the following functions:

- The SOURCEENCODING value is used to interpret the source data. This overrides any encoding value that is set in the data definition or the data. These values are typically set using the encoding= value in the XML data or the syntax id in EDIFACT data.
- The ENCODETARGET value is used to generate the target data. This overrides any encoding value that is set in the data definition (for flat file data), or from the EncodeTarget map property.

The SOURCEENCODING keyword has a special substitution value that enables WebSphere Data Interchange to use the value from the MQ header. If the SOURCEENCODING value is set to MQCCSID, the parser converts this value as follows:

- If the incoming message has an RFH2 header, it gets the CodedCharSetId value from the RFH2 header.

- If the incoming message has no RFH2 header, but has an MQMD header, it gets the CodedCharSetId value from the MQMD header.
- If a CodedCharSetId value is found in either the MQMD or RFH2 header, the parser converts the CCSID to an encoding name as follows:
 1. The parser looks up the value in the translate table CCS2ENC.

Note: This value is an integer value. The parser needs to convert the integer value to a character string for the table lookup.

For example, the value for RFH2 CCSID value is 1208. The translate table that the value 1208 is mapped to the Unicode value of UTF-8, then the value UTF-8 is substituted for the SOURCEENCODE value.
 2. If the value is not found, the encoding name “ibm-nnnn” is used, where “nnnn” is the CCSID.

For example, if the CCSID value is 1208 and it is not in the CCS2ENC table, the encoding name ibm-1208 is used.
 3. The converted value is used as the SOURCEENCODE value and an informational message will be issued to indicate which encoding name was used.
- If the message does not have an RFH2 or MQMD header, a warning message is issued and the source encoding is processed as if the SOURCEENCODE value was not specified, that is, it is detected from the data or the default encoding.

IGNOREBOM

This keyword prevents WebSphere Data Interchange from misinterpreting binary fields at the beginning of a data format record as a byte-order mark (BOM). A Byte Order Mark (BOM) is the character as a marker to indicate that text is encoded in UTF-8. Valid values are:

- | | |
|----------|---|
| Y | Ignore any byte-order mark, just treat as part of data. |
| N | Process the byte-order mark as encoding information, not part of data. This is the default. |

Note: This keyword only applies to data formats. It does not apply to EDI or XML, because these syntaxes cannot begin with binary information. If these bytes appear at the beginning of an EDI or XML file, the only valid way to interpret them would be as a byte-order mark.

Byte-order mark support

WebSphere Data Interchange Version 3.3 supports Byte Order Mark (BOM). BOM is the character at code point U+FEFF (zero-width no-break space). When that character is used as a marker to indicate that text is encoded in UTF-8, UTF-16 or UTF-32.

Table 48 on page 318 describes the BOMs recognized by WebSphere Data Interchange for input messages (all syntaxes)

Handling international characters

Table 48. WebSphere Data Interchange supported BOMs

Byte order mark	Description
00 00 FE FF	UCS-4, big-endian machine (1234 order)
FF FE 00 00	UCS-4, little-endian machine (4321 order)
FE FF xx xx	UTF-16, big-endian
FF FE xx xx	UTF-16, little-endian
EF BB BF	UTF-8

Notes:

1. Endian refers to sequencing methods used in a one-dimensional system (such as writing or computer memory). The two main types of endianness are known as big-endian (big units first) and little-endian (little units first).
2. Big endian is the most significant byte (MSB) stored at the memory location with the lowest address.
3. Little-endian is the least significant byte (LSB) first.

If any of the byte sequences in Table 48 are found at the beginning of an input file, the encoding for each of the documents in the input file will be set to the encoding specified by the BOM.

The priority for determining the encoding is:

1. The SOURCEENCODE keyword, if specified.
2. The byte-order mark at the start of the file, if present.
3. The encoding determined based on the data, if appropriate (code page in Data format definition, XML encoding= value, or EDIFACT syntax id).
4. The default system code page.

A new PERFORM keywords exist in WebSphere Data Interchange for the prevention of misinterpreting binary fields at the beginning of a Data Format record as a byte-order mark. See “PERFORM keywords for international data” on page 316.

For output messages (all syntaxes) a new target document property, ByteOrderMark, is available.

If this property is set to Y, a byte-order mark (based on the output encoding type) is added when a new file is created and the EncodeTarget property is set to one of the encodings described Table 48. When appending to an existing file, no byte-order mark is added.

If the ByteOrderMark property is N, byte order marks will not be added. If the underlying code page conversion libraries set the byte-order mark, it is removed. If the ByteOrderMark is not specified, then the output will continue to appear as it is today. That is, if the code page conversion libraries add the BOM, it is left as part of the message. If the code page conversion libraries do not add the BOM, WebSphere Data Interchange does not insert it.

Examples

The following scenarios highlight some of the international tasks that WebSphere Data Interchange handles.

Scenario 1: WebSphere Data Interchange in a worldwide data center

You are using WebSphere Data Interchange in a worldwide data center which handles transactions from many different countries. The data center is translating XML, EDI, and data format data from many different countries. Thus the transactions contain a wide variety of characters. In addition to the Latin characters A–Z, a–z, and 0–9, the data also include characters from many other languages, including Chinese, Japanese, Cyrillic, Arabic, and Greek.

You need to use any of the characters in your data as part of the Data Transformation processing, including:

- Trading partner profiles
- Translation tables and code lists
- Mapping commands
- XML tag names

Example: Transforming data containing international characters

To transform the input that contains characters from different international alphabets, you indicate the source and target encodings the same way as you did in the previous version of WebSphere Data Interchange. That is, by using XML autodetection logic, having EDI use the EDISNTX table, and ADF use the code page definition.

Or, you can use the new SOURCEENCODE keyword. Using this keyword enables you to use all WebSphere Data Interchange Data Transformation functions, regardless of the source and target encodings and the characters being used.

Scenario 2: Exporting data from one database and importing it into another

The following use cases show how WebSphere Data Interchange handles international characters during the importing and exporting of data with the client and server.

Example 1: Exporting and importing using the client

Export the data from the client as usual. Data exports in UTF-8 format, which can represent any character in virtually any language.

You import the data as usual the database. WebSphere Data Interchange recognizes that the import file is UTF-8 and interprets it appropriately.

Example 2: Exporting and importing using the server

You can export data from the server as usual. The data exports in UTF-8 format, which can represent any character in virtually any language. You can then import the data as usual into the database. The server recognizes that the import file is UTF-8 and interprets it appropriately.

Handling international characters

Note: Import files are not readable on a z/OS server, but can be downloaded in binary format and viewed on a Windows system.

Example 3: Exporting using the client, importing using the server

You can export data from the client as usual. The data is exported in UTF-8 format, which can represent any character in virtually any language.

You can then upload the data in binary format to the server and import the data as usual into the database. The server recognizes that the import file is UTF-8 and interprets it appropriately.

Example 4: Exporting using the server, importing using the client

You can export data from the server as usual. The data is exported in UTF-8 format, which can represent any character in virtually any language. You then download the data in binary format to the WebSphere Data Interchange Client and import the data as usual into the database. The WebSphere Data Interchange Client recognizes that the import file is UTF-8 and interprets it appropriately.

Note: Import files are not readable on a z/OS server, but can be downloaded in binary format and viewed on a Windows system.

Scenario 3: Migrating from a previous version of WebSphere Data Interchange

You have existing version of WebSphere Data Interchange and want to:

- Migrate to version 3.3.
- Continue using existing Send, Receive and Data Transformation maps.

Since the earlier version of WebSphere Data Interchange did not support Unicode, the EIF file and all of the migrated data is in the local default code page.

Example 1: Importing data with the client (either for release migration or individual EIF files)

You can import the data as usual into the database. WebSphere Data Interchange recognizes that the import file is in the local code page and interprets it appropriately.

Note: Only migration of WebSphere Data Interchange 3.2 maps (both send and receive and Data Transformation) is supported when importing using the client. The migration of WebSphere Data Interchange 3.1 map databases is not supported by the client.

Example 2: Importing data with the server (either for release migration or individual EIF files)

You can import the data as usual into the database. The server recognizes that the import file is in the local code page and interprets it appropriately.

Note: If the import file contains WebSphere Data Interchange 3.1 maps, they are converted to WebSphere Data Interchange 3.3 format send and Receive maps during the import process.

Appendix D. DTD Conversion Utility

The DTD Conversion Utility is used to create the EDI standard from an XML DTD. The EDI standard is a representation of the XML data using an EDI-type syntax that is understood by WebSphere Data Interchange. The generated EDI standard has a structure similar to the XML document.

Note: XML is handled natively in WebSphere Data Interchange Version 3.3. It is not necessary to use the DTD Conversion Utility with WebSphere Data Interchange Version 3.2 and later unless you are updating a DTD that was converted in Version 3.1. When using XML in WebSphere Data Interchange Version 3.2 and later, the DTD is imported directly into the WebSphere Data Interchange Client.

Converting a DTD

1. Start the DTD Conversion Utility by selecting the corresponding shortcut or by double-clicking on the DTDCONVERT.EXE file located in the Data Interchange install directory.

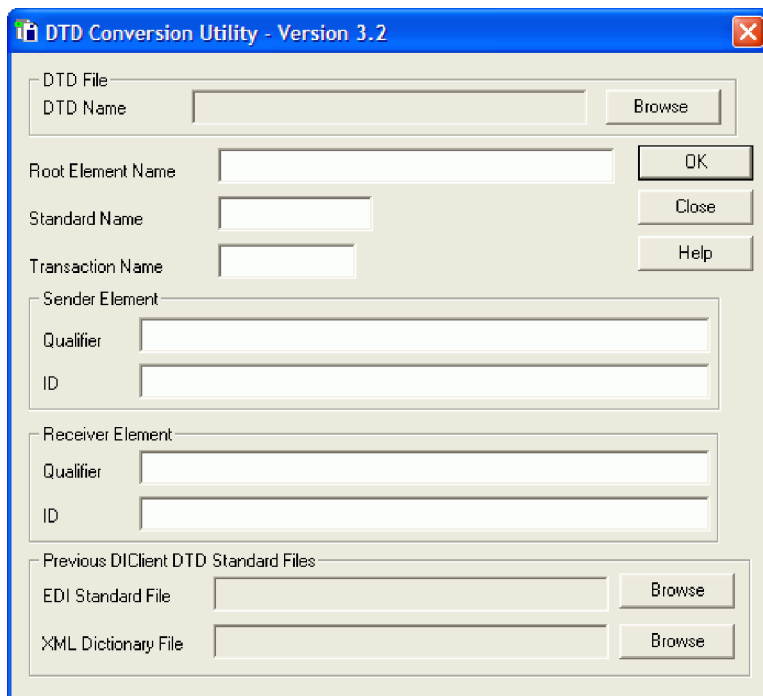


Figure 31. DTD Conversion Utility

- Click Browse to locate the XML DTD file to be converted by the utility. The DTD Name is a required field. For information about DTD file name resolution, see “Resolving DTD file names” on page 328.

Note: If the DTD contains a reference to an external DTD and you want the parser to process the external DTD along with the XML data, then you must upload the DTD file to the server. See “Processing external DTDs” on page 327 for more information about processing an external DTD.

- Specify the Root Element Name. This field is required.
The root element is an element that contains all other elements of the document. The maximum length is 64 characters. Only valid XML element name characters are permitted in this field. The valid characters for XML elements include letters, numbers, and the following special characters: dot (.), dash (-), underscore (_), and colon (:).
- Complete the rest of the optional fields. Table 49 describes the valid values for each field.

Table 49. DTD Conversion Utility field descriptions

Field	Description
Standard Name	The name of the EDI standard to be generated from this DTD. If not specified, the first 8 alphanumeric characters of the root element name are used. The maximum length is 8 characters. Only letters and numbers are permitted in this field.
Transaction Name	The name of the EDI standard transaction associated with this DTD. If not specified, the first 6 characters of the root element name are used. The maximum length is 6 characters. Only valid XML element name characters are permitted in this field. The valid characters for XML elements include letters, numbers, and the following special characters: dot (.), dash (-), underscore (_), and colon (:).
Qualifier (Sender Element pane)	<p>The XML element name, attribute, or path that defines the sender qualifier for the standard. The maximum length is 64 characters. Only valid XML element name characters, and the slash (/) and dollar sign (\$) characters to indicate path or attribute names, are permitted in this field. The valid characters for XML element names include letters, numbers, and the following special characters: dot (.), dash (-), underscore (_), and colon (:).</p> <p>For inbound processing, the value from the sender qualifier is used as the interchange sender qualifier. If the Sender Qualifier element is not specified, or if the specified element, attribute or path is not found in the XML data, the value ZZ is used as the default sender qualifier.</p> <p>For outbound processing, this field is not used, and the interchange sender qualifier is ZZ.</p> <p>For more information, see “Specifying Sender and Receiver Information” on page 332.</p>

Table 49. DTD Conversion Utility field descriptions (continued)

Field	Description
ID (Sender Element pane)	<p>The XML element name, attribute, or path that defines the sender ID for the standard. The maximum length is 64 characters. Only valid XML element name characters, and the slash (/) and dollar sign (\$) characters to indicate path or attribute names, are permitted in this field. The valid characters for XML element names include letters, numbers, and the following special characters: dot (.), dash (-), underscore (_), and colon (:).</p> <p>For inbound processing, the value from the sender ID element is used as the interchange sender. If the Sender ID element is not specified, or if the specified element, attribute or path is not found in the XML data, the value XMLPROC is used as the default sender ID.</p> <p>For outbound processing, this field is not used, and the interchange sender ID is XMLPROC.</p> <p>For more information, see “Specifying Sender and Receiver Information” on page 332.</p>
Qualifier (Receiver Element pane)	<p>The XML element name, attribute, or path that defines the receiver qualifier for the standard. The maximum length is 64 characters. Only valid XML element name characters, and the slash (/) and dollar sign (\$) characters to indicate path or attribute names, are permitted in this field. The valid characters for XML element names include letters, numbers, and the following special characters: dot (.), dash (-), underscore (_), and colon (:).</p> <p>For inbound processing, the value from the receiver qualifier is used as the interchange receiver qualifier. If the Receiver Qualifier element is not specified, or if the specified element, attribute or path is not found in the XML data, the value ZZ is used as the default receiver qualifier.</p> <p>For outbound processing, this field is not used, and the interchange receiver qualifier is ZZ.</p> <p>For more information, see “Specifying Sender and Receiver Information” on page 332.</p>

Table 49. DTD Conversion Utility field descriptions (continued)

Field	Description
ID (Receiver Element pane)	<p>The XML element name, attribute, or path that defines the receiver ID for the standard. The maximum length is 64 characters. Only valid XML element name characters, and the slash (/) and dollar sign (\$) characters to indicate path or attribute names, are permitted in this field. The valid characters for XML element names include letters, numbers, and the following special characters: dot (.), dash (-), underscore (_), and colon (:).</p> <p>For inbound processing, the value from the receiver ID element is used as the interchange receiver. If the Receiver ID element is not specified, or if the specified element, attribute or path is not found in the XML data, the value XMLRCVR is used as the default receiver ID.</p> <p>For outbound processing, this field is not used, and the interchange receiver ID is XMLRCVR.</p> <p>For more information, see “Specifying Sender and Receiver Information” on page 332.</p>
EDI Standard File	<p>If converting a DTD that was previously converted using the DTD Conversion Utility, specify the name of the WebSphere Data Interchange export file that contains the EDI standard created in the previous conversion. You can use the import file generated in the previous conversion of this DTD or you can export the EDI standard from WebSphere Data Interchange. The export file must be in tagged export format.</p> <p>This field is used in conjunction with the XML Dictionary File field.</p> <p>See “DTD Conversion Utility output” on page 325 for additional information about the import file generated when running the DTD Conversion Utility.</p>
XML Dictionary File	<p>If converting a DTD that was previously converted using the DTD Conversion Utility, this field is used to identify the name of the original XML dictionary file generated from the previous conversion. You must reconvert a DTD if the DTD is changed. For more information about reconverting the DTD see “Updating a previously converted DTD” on page 327.</p> <p>This field is required if the EDI Standard file field is used. If the EDI Standard File field is not specified, then this field is ignored. This field identifies the name of the original XML dictionary file generated from the DTD Conversion Utility using the original XML DTD.</p> <p>See “DTD Conversion Utility output” on page 325 for additional information about the XML dictionary file generated when running the DTD Conversion Utility.</p>

5. Click OK to begin converting the DTD.

During the conversion processing, a status window is opens. The status window opens messages pertinent to the conversion, including identifying the names of the two output files produced by the utility. You can close the status window clicking Close.

6. After the conversion of a DTD has been completed, you can convert another DTD by specifying the necessary information in the DTD Conversion Utility. Once you have completed all conversions, close the utility by clicking Close.

Two output files are produce for each DTD that is converted. The first file is a common WebSphere Data Interchange import file. It contains an EDI standard the correlates to the DTD. The second file is a proprietary XML dictionary file that associates XML element and attribute names with the EDI standard segments and data elements. For more information about the DTD Utility output files, see “DTD Conversion Utility output.”

If the DTD is ever changed, you need to reconvert the DTD. See “Updating a previously converted DTD” on page 327 for more information.

DTD Conversion Utility output

In WebSphere Data Interchange Version 3.1, implementing XML processing using WebSphere Data Interchange required the use of the DTD Conversion Utility. The DTD Conversion Utility is a standalone program. It reads an XML DTD and generates an EDI standard representation of the XML structure. It parses the DTD file selected by the user. From the DTD file and other user input, the DTD Conversion Utility generates two output files:

- WebSphere Data Interchange Client import file:

The WebSphere Data Interchange Client import file is used to import the new EDI standard into WebSphere Data Interchange Client. The EDI standard is used to define the DTD in terms of an EDI standard. The filename is in the format *stdname*.EIF, where *stdname* is the standard name you specify. If you do not specify a standard name, the standard name is taken from the first 8 characters of the root element.

- XML dictionary file:

The XML dictionary is used for runtime processing. It is used to correlate the XML element and attribute names with EDI standard segments and data elements. The filename is in the format *stdname*.DIC, where *stdname* is the standard name you specify. If you do not specify a standard name, the standard name is taken from the first 8 characters of the root element. This file is uploaded to the server and placed either in a PDS member or an HFS file. For more information on the naming of the server file for the XML dictionary, see “Resolving the XML dictionary file name” on page 326.

You also have the original DTD file. The DTD file is optional for runtime processing. It can be used to validate the inbound or outbound XML data or specify default XML attribute values. Whether WebSphere Data Interchange uses the DTD or not depends on the validation level for the transaction. If you want WebSphere Data Interchange to process the DTD during the XML parsing, this file is uploaded to your server and placed

in a PDS member or HFS file. For information on naming this file, see “Processing external DTDs” on page 327. The DTD and XML dictionary files can be considered library type files for XML processing.

XML Dictionary File

If converting a DTD that was previously converted using the DTD Conversion Utility, this field is used to identify the name of the original XML dictionary file generated from the previous conversion. You need to reconvert a DTD if the DTD is changed. This field is required if the EDI Standard file field is used. If the EDI Standard File field is not specified, then this field is ignored. This field identifies the name of the original XML dictionary file generated from the DTD Conversion Utility using the original XML DTD.

Resolving the XML dictionary file name

The XML dictionary is used by the XML processor to correlate the XML elements and attributes with the EDI standard segments and data elements. It is one of the two files that are generated by the DTD Conversion Utility when you convert a DTD. The file is saved in the same directory as the DTD being converted. The filename is in the format *standard.DIC*, where *standard* is the standard name being generated.

This file must be uploaded to the server so the XML processor can access it during translation. It is uploaded using text mode, because it is generated as an ASCII text file. It can be stored on the server as either a PDS member, or as an HFS file. If it is stored as a PDS member, the member name must be the same as the standard name. If it is stored as an HFS file, the filename must be in the format */path/standard.DIC*, where *path* is the HFS directory for the file, and *standard* is the standard name (in uppercase).

The XMLDICT keyword on the PERFORM command is used to specify the location of this file for the XML processor. If the XMLDICT value starts with a slash (/) character, the dictionary is assumed to be an HFS file, and the XMLDICT value specifies the directory. If the XMLDICT value does not start with a slash (/) character, the dictionary is assumed to be a PDS member, and the XMLDICT value specifies the fully qualified PDS name.

For inbound processing, the XMLSTDID keyword identifies the EDI standard (dictionary) name to be used. If the XMLSTDID keyword is not used, the first 8 alphanumeric characters of the root element in the XML data is used as the EDI standard (dictionary) name.

For outbound processing, the dictionary name is defined in the map (the standard dictionary name used for the map).

For example, if the dictionary name is CXML and is located in the PDS EDI.XML.DICT then the dictionary file would be in EDI.XML.DICT(CXML). Your PERFORM command would look something like this:

```
PERFORM TRANSLATE AND ENVELOPE WHERE XMLDICT(EDI.XML.DICT)
PERFORM DEENVELOPE AND TRANSLATE WHERE XMLSTDID(CXML) XMLDICT(EDI.XML.DICT)
```

If the dictionary name CXML is located in HFS directory */u/ediuser*, it would be in file */u/ediuser/CXML.DIC*. The PERFORM command would look something like:


```
PERFORM TRANSLATE AND ENVELOPE WHERE XMLDICT(/u/ediuser)
PERFORM DEENVELOPE AND TRANSLATE WHERE XMLSTDID(CXML) XMLDICT(/u/ediuser)
```

Note: For HFS files, the path and filenames are case sensitive.

Updating a previously converted DTD

The mappings of WebSphere Data Interchange Client are based on the segment position numbers and element position numbers in the EDI Standard. Every time a DTD is changed, the position numbers get adjusted in the generated EDI standard. Thus mappings can be lost after the DTD is converted and re-imported into WebSphere Data Interchange Client. To address the problem with losing mappings each time an XML DTD is reconverted using the XML DTD Conversion Utility and then re-imported into WebSphere Data Interchange Client, the EDI Standard File and XML Dictionary File fields are available and must be provided when a DTD is reconverted. Click Browse to locate and identify the previous EDI standard import file and XML dictionary file that are used as input to the converter.

Using the reconvert process, the XML DTD Conversion Utility attempts to merge the DTD changes into the existing EDI standard so that the mapping losses are minimized. The output is an updated WebSphere Data Interchange EDI standard. The output EDI standard is named `standard_upd.eif`.

At most, nine consecutive segments can be added in between any two segments in the original, unchanged DTD file. The sequence numbers in an EDI standard are normally in intervals of 10. For example 20 , 30, 40 and so on. The DTD Conversion Utility adjusts a maximum of nine elements between any two segments without affecting the following mappings. For example, between segments 20 and 30 the new segments have sequence numbers 21, 22, 23 an so on.

The `stdadjust.log` message file contains the list of EDI standard segments and elements that lose mappings after the output EDI standard is imported. All the segments within a loop with lost mappings also lose their mappings. The same is true with elements within a segment that loses its mappings.

Processing external DTDs

External DTDs can be parsed along with the XML data. The DTD can be used to validate the XML data. If you want the parser to process an external DTD along with the XML data, then you must upload the DTD file to the server. The file is stored as either a PDS member or an HFS file.

For outbound processing (data format to XML), the DTD is uploaded as text. For inbound processing (XML to data format), if you specify `XMLEBCDIC(Y)`, the DTD file is uploaded as text. If you specify `XMLEBCDIC(N)`, then the DTD file must contain an XML declaration (“`<?xml...>`”) that includes the appropriate encoding type for the file. See “Encoding considerations” on page 336 for more information.

Resolving DTD file names

If DTD processing is to be done as part of the XML parsing (based on the validation level), then the following processing takes place whenever the parser finds a reference to an external DTD file:

1. When an external DTD reference is found in the XML data, all path information (such as a URL path or file path information) is removed from the DTD name, leaving only a base DTD name.
2. If a DTD alias file (DD:DTDALIAS) is allocated, then the base name is first looked up in the alias file. See “Using a DTD alias file” on page 329 for more information.
3. If the base name is found in the alias file, then the alias name is combined with the XMLDTDS value to determine the filename of the DTD file. The search for the base name in the alias file is not case sensitive.
4. If no alias is found (or the DTDALIAS file is not allocated), then the base DTD name is combined with the XMLDTDS value to determine the filename of the DTD file. This DTD file is used by the XML parser in place of the URL specified on the DOCTYPE declaration.

Normally, the XML processor takes the DTD name (without file or URL path information) and combines it with the XMLDTDS value provided on the PERFORM command. If the XMLDTDS value starts with a slash (/), then it is assumed to be an HFS path, and the base name is appended to the path. Otherwise, the XMLDTDS value is assumed to be a PDS, and the first 8 characters of the base name (minus the extension) are assumed to be the member name.

For outbound processing, the DTD name is in the XML dictionary created by the DTD Conversion Utility. If you use validation level 2, the DTD name is retrieved from the XML dictionary and used to create the default DOCTYPE declaration. If you use validation level 1, the DTD name is not included in the default DOCTYPE declaration, but if you override the default prolog using the DIPROLOG variable and specify a DTD, the DTD name is processed. If you use validation level 0, outbound XML data is not validated using a DTD.

Example of DTD file resolution

If the XML data contains the following DOCTYPE declaration:

```
<!DOCTYPE PurchaseOrder SYSTEM "http://xyz.org/xml/dtds/MyPO.dtd">
```

The DTD name would be resolved as follows:

1. Remove the URL path information, leaving MyPO.dtd as the base name.
2. Look up MyPO.dtd in the DTDALIAS file. For this example, assume it is not found.
3. If the XMLDTDS value starts with a slash (/), then the base name is appended to the XMLDTDS value. If XMLDTDS(/u/ediuser/mydtds) is specified, then the DTD file name would be /u/ediuser/mydtds/MyPO.dtd.
4. If the XMLDTDS value does not start with a slash (/), then the base name (without the extension) is assumed to be a PDS name. If XMLDTDS(EDIUSER.MYDTDS) is specified, then the DTD file name would be EDIUSER.MYDTDS(MYPO).

Using a DTD alias file

This behavior can cause conflicts in certain cases, such as with long DTD names or DTD names that differ only in the extension. To resolve conflicts caused by long DTD names, use the DTDALIAS file to specify an alias for a DTD name. For example, if you are keeping your DTD files in a PDS, but you have two DTD files: LongDTDName1.dtd and LongDTDName2.dtd using the first 8 characters would yield the same member name for both: LONGDTDN. The DTDALIAS file enables you to specify different member names for each of these.

The format is the DTD (base) name followed by one or more blanks, followed by the alias name. For example:

```
longtdname1.dtd  LONGN1
longtdname2.dtd  LONGN2
```

Note: HFS file names are case sensitive. PDS member names are not case sensitive.

As an example, if you used the DTDALIAS file, specified the parameter XMLDTDS (EDIUSER.DTDS) on your PERFORM command, and your XML data contained the following DOCTYPE declaration:

```
<!DOCTYPE po SYSTEM "http://xyz.org/xml/dtlds/LongDTDName1.dtd">
```

The parser would resolve the DTD as follows:

1. Remove the URL path information from the DTD name, resulting in a base name of LongDTDName1.dtd.
2. Look up this base name in the DTDALIAS file. Since this search is not case sensitive, it would find the first entry, and result in an alias name of LONGN1.
3. Combine the alias name with the XMLDTDS value. Since EDIUSER.DTDS does not start with a slash (/), it is assumed to be a PDS. Therefore, the member EDIUSER.DTDS(LONGN1) is processed as the DTD file for this XML document.

Inbound Translation Process (XML to Data Format)

The inbound translation process is similar to the current EDI to data format (ADF) process flow. The WebSphere Data Interchange Utility identifies the XML data as input using the PERFORM keyword XML(Y/N). If XML(Y) is specified, the Utility calls the XML processor service to convert the XML data into an EDI standard format, including an EDIFACT envelope, and writes it to the XML work file XMLWORK. The Utility continues the inbound process flow as normal through the translation process, and passes the XMLWORK file as the FILEID parameter to translation services.

The level of XML validation is determined by the VALIDATE keyword on the Utility PERFORM command.

- | | |
|---|--|
| 0 | Indicates that external DTD references are ignored. |
| 1 | Indicates the DTD specified on the DOCTYPE declaration is processed, but no DTD validation is done. This is the default. |
| 2 | Indicates full validation against DTD. |

All XML data must be well-formed to be processed. Validation level 1 and 2 requires the XMLDTD keyword to be used on the Utility PERFORM commands and that the DTD files are on the server system.

XML processor errors are written to file XMLERR and signal the Utility to also log and write an error message to the Audit trail report. The XML data containing errors is written to the XMLEXCP file.

XML processor trace messages are written to the XMLTRC file if allocated. This file is for WebSphere Data Interchange development use only. This file is typically allocated when doing problem determination. Allocating this file during normal translation reduces performance.

Document store and management reporting can be used with the current implementation. XML data is represented in EDI standard format. Since the Document store is usable and the XML data is stored in EDI standard format, other inbound commands are supported. The interchange control numbers are generated from the system clock to avoid duplicate envelopes.

Receiving XML data using VAN (inbound)

If you are using XMLBDCIC(Y), you can also use the XMLSEGINP(Y/N) keyword to control whether the record boundaries on your input XML data are treated as line breaks or ignored.

- If you use XMLSEGINP(Y), then a newline character is assumed at the end of each input record in the XML data. This will give more information for any error messages generated by the XML parser, because it will include more accurate line and column information. However, this requires that the record boundaries only occur where white space characters are valid, such as between elements.
- If you use XMLSEGINP(N), then record breaks are ignored and the data is treated as a continuous stream of characters. This enables record breaks to occur anywhere in the data. However, any parser errors will not accurately show the line number, because the entire document is treated as if it were a single line.

For XMLBDCIC(N), the record boundaries are always ignored, and line numbers are based on the carriage-return or newline characters that appear in the data.

The Interchange Sender ID and Qualifier will default to XMLPROC and ZZ if no sender ID is present in the XML data. This means a Trading Partner profile and Trading Partner receive usage for sender ID XMLPROC must be defined and attached to the XML map. The interchange and transaction control numbers will be based on the system clock to avoid duplicate envelopes in Transaction store.

The PERFORM RECEIVE command will receive EDI data only. It is a two step process to receive XML data using a Value-added Network (VAN):

```
PERFORM RECVFILE
```

```
PERFORM DEENVELOPE AND TRANSLATE.
```

It is also a two step process to translate from XML to an EDI standard such as X12 or EDIFACT. First you use this XML processor to translate from XML to application data. Then you translate the application data to EDI.

Outbound Translation Process (Data Format to XML)

The outbound process is similar to the current data format (ADF) to EDI process flow. The translation process determines the map to use with the current trading partner usage lookup. An XML map is identified using envelope type L for the EDI standard transaction. If the envelope type is L, each new transaction forces an interchange break to build all EDIFACT envelope segments. The XML processor converts the EDI standard data to XML data.

XML validation is determined by the trading partner usage overrides:

- 0** Indicates that no checking is done on the outbound XML data.
- 1** Verifies that the outbound XML data is well-formed. If a DTD is specified in the DIPROLOG variable, it is processed for default attributes, parameter entity references, and so on. The XML data is not checked against the DTD for validity.
- 2** Validates the outbound XML data against the DTD, in addition to verifying that it is well-formed. This requires the XMLDTD keyword to be used on the Utility PERFORM commands.

Note: Validation of the outbound XML data causes additional processing, so it can impact performance.

The Trading Partner Profile definition flag (segmented output = Y) causes XML data to be written out in an easy to read format with line breaks and indentation. If segmented output = N, the data is written as a continuous stream, with no blanks or line breaks inserted. For more information about Trading Partner setup, see “Identifying trading partners” on page 334.

XML processor errors and status messages are written to a file XMLERR. If an XML processing error occurs, the processor signals the Utility to also log and write an error message to the Audit trail report. If the XML data is generated by the XML processor, but an error is found during the validation, the invalid XML data is written to the XMLEXCP file. Form more information about processing errors, see “XML Processor Messages and Codes” on page 344.

XML processor trace messages are written out to the XMLTRC file if allocated. This file is for WebSphere Data Interchange development use only. It is typically allocated only when doing problem determination. Allocating this file during normal translation reduces performance.

Document store and management reporting can be used with current implementation. XML data is represented in EDI standard format. Since the Document store is usable and the XML data is stored in EDI standard format, other outbound commands are

supported. The interchange sender and qualifier are XMLPROC and ZZ. Interchange control numbers are updated based on trading partner similar to current EDI processing.

Sending XML data using VAN (outbound)

The PERFORM SEND command sends EDI data only. It is a two step process to send XML data using Value-added Network (VAN):

```
PERFORM TRANSLATE AND ENVELOPE
```

```
PERFORM SENDFILE
```

The mapping variable DIPROLOG enables you to map an override for the default XML prolog generated by the XML processor. You can use this if you want to customize the XML prolog information, such as the encoding type or the DOCTYPE declaration. The maximum number of characters that can be saved is approximately 900.

To map from an EDI standard such as X12 or EDIFACT to XML format, there are two options:

- You can map the EDI data to a data format (ADF) using server or WebSphere Data Interchange Client mapping. With this option you need to map XML prolog, starting, and ending tags as well as control any formatting wanted such as line breaks and indentation.
- You can use the XML processor to translate from EDI to application data followed by application data to XML translation.

Specifying Sender and Receiver Information

To identify the sender and receiver ID and qualifier fields using the DTD Conversion Utility, you enter the name of the XML element that contains the value. For example, if the XML data contains the following:

```
<Header>  
  <From>SenderName</From>  
  <To>ReceiverName</To>  
</Header>
```

You would just enter the values From and To in the Sender ID and Receiver ID fields. This example does not have any qualifiers, so you would leave those blank.

Note: These fields, like all XML elements names, are case sensitive.

Sometimes the sender and receiver information is in an attribute. The format to specify an attribute is elementName\$attributeName. For example, if the XML data contains:

```
<Header>  
  <From type="DUNS">SenderName</From>  
  <To type="DUNS">ReceiverName</To>  
</Header>
```

You would specify the sender and receiver IDs as in the first example. For the sender and receiver qualifiers you would specify:

Sender qualifier: "From\$type"

Receiver qualifier: "To\$type"

Attribute names, like element names, are case sensitive.

In some cases, the sender and receiver information is specified not just by a unique element, but also by the context in which the element occurs. For example, the information can reflect information from the parent or ancestor elements. Take the following XML as an example:

```
<Header>
  <From>
    <Credential domain="DUNS">
      <Identity>942888711</Identity> <!-- Sender ID -->
    </Credential>
  </From>
  <To>
    <Credential domain="Name">
      <Identity>XMLTEST1</Identity> <!-- Receiver ID -->
    </Credential>
  </To>
</Header>
```

In this example the sender and receiver IDs are both located in Identity elements, and the qualifiers occur in the domain attribute of the Credential elements. But, the Credential and Identity elements occur within both the From and To elements so you need to identify the path to sender and receiver. You only need to include as much path information as required to uniquely identify the element. You can use the To structure for the receiver ID and the From structure for the sender ID. In the DTD Conversion Utility you enter the following:

Sender ID element	From/Credential/Identity
Sender qualifier element	From/Credential\$domain
Receiver ID element	To/Credential/Identity
Receiver qualifier element	To/Credential\$domain

Other notes on specifying the sender and receiver information:

- The maximum path length for each sender and receiver element field is 64 characters.
- If the values in the XML data for the sender and receiver ID are longer than 35 characters, they are truncated to 35 characters
- If the values in the XML data for the sender and receiver qualifier are longer than 4 characters, they are truncated to 4 characters

Identifying trading partners

Since the application data is converted to EDI data with an EDIFACT envelope as the first step, trading partner identification works the same as current EDI processing. The internal trading partner ID identified on the RAW data format or the WebSphere Data Interchange C record is used along with the data format ID to locate the map.

For XML processing, the interchange sender ID and qualifier is XMLPROC and ZZ. The receiver ID is populated using the trading partner profile information similar to traditional processing. A minimal trading partner setup can be used, see *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01 for more information about minimal trading partners.

For XML processing, the setup is the same as regular EDI processing. You set up the trading partner profile and attach a trading partner usage to the map. Some trading partner usage fields do not apply to XML data such as application sender/receiver, functional acknowledgement fields, encryption, security, and authentication. These EDI type fields are not used for XML processing and can produce unexpected results. The envelope type field on the trading partner usage override must be 'L' for XML.

The envelope type indicates this is an XML trading partner. There is no envelope profile for XML.

PERFORM Commands and Keywords

The following PERFORM commands can be used to process outbound XML data:

```
PERFORM TRANSLATE TO STANDARD
PERFORM ENVELOPE
PERFORM TRANSLATE AND ENVELOPE
```

The following keywords are used on PERFORM commands to control how the outbound XML data is processed:

XMLDTDS

Required for XML DTD processing. Identifies the PDS or HFS path where the XML DTD members are located. Maximum length is 64. See "Resolving DTD file names" on page 328 for more information.

XMLDICT

Required for XML processing. Identifies the PDS or HFS path where the XML dictionary files generated by the DTD Conversion Utility are located. Maximum length is 64.

XML Required for XML processing. Default is N.

Y Indicates that input data is XML.

N indicates normal processing (no XML processing).

XMLVALIDATE

Optional. All XML data must be well-formed to be processed. The default is 1.

- 0** Indicates that external DTD references are be ignored.
- 1** Indicates that if a DTD is specified on the DOCTYPE declaration, it is processed (for example. default attributes, entity references, and so on), but no DTD validation is done.
- 2** Indicates full validation against DTD.

XMLSTDID

Optional for XML processing. The standard ID that was created with the DTD Conversion Utility. If this keyword is used, the specified standard ID is used to convert each XML document to a corresponding EDI standard envelope and transaction. If this keyword is not specified, then the first 8 alphanumeric characters of the root element name are used as the standard ID for each XML document. The maximum length is 8 characters.

MULTIDOCs

Optional for XML processing. Default is N.

- Y** Indicates the XML input file contains multiple documents. If Y is specified, the input message must be in EBCDIC and each document must begin with an XML declaration (<?xml...).
- N** Indicates the XML input file contains one document.

XMLLEBCDIC

Optional for XML processing. Default is Y.

- Y** Indicates the XML data is to be interpreted as EBCDIC.
- N** Indicates that the encoding on XML declaration is used. See the “Encoding considerations” on page 336 for more detail

XMLSEGINP

Optional for XML processing. Default is Y.

- Y** Indicates that the record boundaries in the input XML data are treated as line breaks.
- N** Indicates that the record boundaries in the input XML data are ignored. This keyword is ignored for XMLLEBCDIC(N). See “Receiving XML data using VAN (inbound)” on page 330 for more detail

XMLDTDS

Required for XML DTD processing. Identifies the PDS or HFS path where the XML DTD members are located. Maximum length is 64. See “Resolving DTD file names” on page 328 for more information.

XMLDICT

Required for XML processing. Identifies the PDS or HFS path where the XML dictionary files generated by the DTD Conversion Utility are located. Maximum length is 64. See “Resolving the XML dictionary file name” on page 326 for more information.

Encoding considerations

The XML processor uses the XML Toolkit for OS/390 to parse the XML data. This parser supports many different character encodings, and follows the XML standards for auto detection of the character encoding. However, this can cause problems when dealing with EBCDIC data.

According to the XML standard, if the XML document is not in UTF-8 (similar to ASCII for most commonly used characters) or UTF-16 format, it must begin with an XML encoding declaration (`<?xml...>`). For EBCDIC data, the `encoding=` attribute must be present and indicate which encoding type is in use. If the XML data was generated as ASCII data, then converted to EBCDIC, it is likely that the `encoding=` attribute would not be added or updated to reflect the EBCDIC conversion.

Additionally, the newline (EBCDIC `x'15'`) character is not recognized as a valid whitespace character by the XML standard. However, on z/OS platforms this is often inserted into files as a record separator by editors, upload applications, and other z/OS applications.

To resolve both of these problems, the XML processor in WebSphere Data Interchange can instruct the parser to override the default (auto detected) encoding type for the XML document with a special encoding type: `"ebcdic-xml-us"`. This causes the parser to ignore the `encoding=` attribute in the XML declaration, and use this special type instead. This encoding type is based on the `ibm1140` code page, and treats any newline characters as a carriage-return (`x'0A'`), which is considered a valid XML whitespace character.

When receiving XML data, the `XMLEBCDIC(Y)` keyword instructs the XML processor to override the default encoding type with the special `ebcdic-xml-us` encoding type. This applies to any external DTDs processed, as well as to the XML data itself. Overriding the code page enables the XML processor to handle XML data and DTDs that are in EBCDIC format, but contain newline characters or does not include the encoding type in the XML declaration (or contains an incorrect encoding type). If `XMLEBCDIC(N)` is specified, then the XML processor determines the encoding type (for both the data and external DTDs) based on the normal XML auto detection rules. You typically use `XMLEBCDIC(N)` if your input XML data is in a format other than EBCDIC. Since the `XMLEBCDIC` value is also used to control the interpretation of the DTD files, using `XMLEBCDIC(N)` also requires that your DTDs contain an XML declaration (`<?xml...>`) with the proper encoding type.

When sending XML data, the XML data is always generated as EBCDIC. The special `ebcdic-xml-us` encoding type checks the XML data and process any external DTDs. No encoding type is specified in the default XML prolog. If you want to send the data in some other format, such as ASCII or UTF-16, the translation must be done outside of WebSphere Data Interchange. In many cases, it can be done by the transport mechanism, such as FTP, that you use to send the data to another system. If you need to specify an `encoding=` attribute in your XML declaration, you can override the default prolog using the `DIPROLOG` mapping variable as described in "Overriding the Default XML Prolog" on page 337.

Additional information about encoding considerations and the XML Toolkit for OS/390 is on the XML Toolkit for OS/390 web site (<http://www.s390.ibm.com/xml/>) under the Usage section.

Overriding the Default XML Prolog

For outbound processing a new mapping keyword, DIPROLOG, is available to override the XML prolog declaration generated during XML processing. DIPROLOG is a special WebSphere Data Interchange variable and can be created using literal data or from your application data. There is no need to issue an &USE with the DIPROLOG variable.

The following is an example of the default prolog without DTD validation:

```
<?xml version="1.0"?>
```

If DTD validation is specified (validation level = 2), a DOCTYPE declaration is added to the prolog which includes the root element and DTD name. The following is a sample default prolog with DTD validation:

```
<?xml version="1.0"?>
<!DOCTYPE cXML SYSTEM "cXML.dtd">
```

To override the generated prolog declaration with literal values, add the following mapping commands using your literal data:

```
&SET DIPROLOG <?xml version="1.0" encoding="iso-8859-1" ?>
&SAVE DIPROLOG,*,* <!DOCTYPE order SYSTEM "order.dtd">
```

Mapping considerations for XML data

Mapping XML data is similar to mapping EDI data using WebSphere Data Interchange Client. To map between a data format and XML, just map the data format fields to the data elements that correspond to the appropriate XML elements and attributes. To map a data format field to an XML element value, map it to the first data element of the segment that corresponds to the XML element. To map a data format field to an attribute of an XML element, map it to the corresponding data element on the segment for the XML element. The full XML element names are provided in the segment descriptions, and the attribute names are included in the data element descriptions.

Most of the segment names are based on the name of the XML element that they represent. However, there are some segments and loops that are not, these segments start with \$SQ and \$CH, and are created by the DTD Conversion Utility to represent certain types of complex DTD element definitions. See “Special Sequences and Choices” on page 340 for more information about \$SQ and \$CH segments.

Many segments do not contain any data elements. These segments are used to indicate the start and end of loops and are automatically qualified with occurrence 1 when you map inner loops using WebSphere Data Interchange Client. The XML and translation processing handles these segments.

Since mapping for XML data is to and from an EDI standard representation of the XML data, all WebSphere Data Interchange mapping functions apply such as Boolean logic,

segment, loop, element qualification, accumulators, application control fields, translation/validation tables, raw data, and C&D application data formats.

EDI Standard Representation of XML Data

There are some special rules used to equate the XML elements and attributes with the standard segments and data elements. The segment and data element descriptions contain the element names and attributes that they correspond to, but a general understanding helps you determine, find and understand these better. The following rules apply:

- Each XML element corresponds to an EDI segment.

The element name is truncated to 6 characters to create the segment ID. If the truncation causes duplicate segment names, then a sequence number is used for the last character(s) to ensure uniqueness. The segment description includes the full XML element name.

- If the XML element is declared as type #PCDATA, ANY, or mixed content (a mixture of #PCDATA and child elements), then the element values correspond to the first data element on the segment.

When receiving these types of elements, all element values within the element are concatenated and placed into the first data element on the segment. The tags and attributes of any nested child elements for types mixed and ANY are discarded (#PCDATA elements do not have nested child elements).

When sending these types of elements, the first data element of the segment is used as the element value.

The description for the first data element includes the name of the XML element, and also its content model, either Any or PCData/Mixed.

- If the XML element is declared as type EMPTY, then the first data element is not used because no element value is permitted.

If the element has attributes (usually the case for EMPTY elements), the first data element on this segment is the special data element UNUSED, which is not mapped. If the element does not have any attributes, then the segment does not have any data elements.

- If the XML element has any attributes, then the attributes correspond to data elements 2-n on the segment.

The description for each data element includes the XML element name, followed by a dollar sign (\$) character, followed by the attribute name. For example, elementName\$attrName.

- If the XML element has child elements, then a loop is defined for it. The first segment of the loop is the segment that corresponds to the XML element as described. The first data element of this segment is the special data element " UNUSED", which is not mapped.
- If the XML element has attributes, they appear as elements 2-n as normal. The start segment does not need to be mapped, unless you want to map attribute values. The last segment of the loop has the same segment id as the first segment, plus a "\$" at the end. The end segment does not need to be mapped. The start and end segments are written if any of the nested segments within the loop contain data.

Sequences

A sequence means that the child elements are defined as a sequence of elements. For example: `<!ELEMENT A (B,C*)>` . This means element A consists of element B, followed by 0 or more occurrences of element C. The XML data might appear something like the following:

```
<A>
  <B>B-Value</B>
  <C>C-value</C>
  <C>C-value2</C>
</A>
```

In this case, element A is defined as a loop, because it has children B and C. Each child in the sequence is defined as a segment within the loop. Each segment is marked as either mandatory or optional, and as either repeating or non-repeating, depending on how the element is defined in the sequence.

In this example, segment B would be mandatory and not repeating, because the DTD states that element B must occur exactly one time in the sequence. Segment C would be defined as optional and repeating, because the DTD states that it occurs 0 or more times. If B or C had other child elements, then they would appear as a nested loop (or loops) within loop A. Loops can be defined as repeating, but are not required to be defined as mandatory.

In this case, it is up to the person performing the mapping to understand the restrictions imposed by the DTD and make sure that some data within loop B is mapped in order to generate valid XML data. Otherwise, the data would not validate properly against the DTD.

Choices

A choice means that the child elements are defined as a choice of elements. For example: `<!ELEMENT A (B|C)>` . This means element A consists of either element B or element C. The XML data might appear something like the following:

```
<A>
  <B>B-Value</B>
</A>
```

or

```
<A>
  <C>C-Value</C>
</A>
```

Here, element A is defined as a loop, because it has children B and C. Each child in the sequence is defined as a segment (or loop) within the loop. Each segment (or loop) in a choice is marked as optional, and not repeating. It is up to the person performing the mapping to understand the structure of the XML data and only map one of the child segments or loops.

Special Sequences and Choices

Some types of XML sequences and choices require special handling to represent them in an EDI type structure. These cases are:

- Repeating sequences
- Repeating choices
- A sequence within a choice
- A choice within a sequence

For each of these, some special segments are generated and inserted in the EDI standard transaction representation. These start with \$CH (for repeating choices or a choice within a sequence) or \$SQ (for repeating sequences or a sequence within a choice). Segments beginning with \$SQ and \$CH have no data elements.

Mapping example

As an example, look at the following XML data structure:

```
<ItemIn quantity="6" lineNumber="1">
  <ItemID>
    <SupplierPartID>pn12345</SupplierPartID>
    <SupplierPartAuxiliaryID>EA</SupplierPartAuxiliaryID>
  </ItemID>
</ItemIn>
<ItemIn quantity="12" lineNumber="2">
  <ItemID>
    <SupplierPartID>823488664ZZZ</SupplierPartID>
    <SupplierPartAuxiliaryID>345602840800</SupplierPartAuxiliaryID>
  </ItemID>
</ItemIn>
```

Now if we look at the sections:

```
<ItemIn quantity="6" lineNumber="1"> Loop with attributes (quantity,lineNumber)
  <ItemID> Loop Segment Element
    <SupplierPartID>pn12345</SupplierPartID>
    <SupplierPartAuxiliaryID> EA</SupplierPartAuxiliaryID>
  </ItemID>End Loop
</ItemIn> End Loop
<ItemIn quantity="12" lineNumber="2">
  <ItemID>
    <SupplierPartID>823488664ZZZ</SupplierPartID>
    <SupplierPartAuxiliaryID>345602840800</SupplierPartAuxiliaryID>
  </ItemID>
</ItemIn>
```

Let's begin with the inner most structures SupplierPartID and SupplierPartAuxiliaryID identified under the heading Segment. These are considered child elements of ItemID. Each child of ItemID, SupplierPartID and SupplierPartAuxiliaryID, has only one child, identified under Element. The SupplierPartID and SupplierPartAuxiliaryID are represented in the EDI standard as segments or repeating segments with one element. The segment ID is generated from the XML element name. The element ID is the segment ID + sequence number. Since SupplierPartID and SupplierPartAuxiliaryID are child elements of ItemID, ItemID is represented in the EDI standard as a loop.

The ItemIn tag is an element with multiple ItemID child elements and also has attributes. Attributes are represented in the EDI standard as data elements with element IDs of segment ID + sequence number and a description of segment ID + '\$' + XML attribute name.

In this example the ItemIn loop is equivalent to a line item loop and can be multiple occurrence qualified using a structure or record. If the SupplierPartID element is mapped without qualifying the ItemID loop, WebSphere Data Interchange Client automatically qualifies the ItemID loop with occurrence 1.

Control String Generation

When mapping is complete, generate the control string using WebSphere Data Interchange Client. If you are using WebSphere Data Interchange Client in standalone mode (not client-server mode), you must export the control string along with any trading partner profile or usage setup and upload/import to the server system.

The EDIFACT envelope standard must be loaded into the server.

Diagnosing Errors

If the DTD Conversion Utility detects a syntax error or issues a warning while trying to parse the DTD, use this section to help you understand any messages.

Understanding DTD parsing

First, a short explanation of how the utility parses the DTD file:

1. First the utility creates short buffer of XML data, containing an XML declaration ("`<?xml...`"), a document type declaration ("`<!DOCTYPE...`"), and an empty root element. It uses the root element and the DTD name that you provide as input to create this XML data.
2. Then the utility passes the XML data to a parser component, which parses the data.
3. If the XML data and the referenced DTD file are well-formed (syntactically correct), then the parser returns information about elements and attributes defined by the DTD. The utility uses this information to generate the XML dictionary file and the WebSphere Data Interchange Client import file.
4. If the data is not syntactically correct, the parser component passes error messages to the utility, which are displayed in the status window.

Parser messages

If the parser detects a syntax error, the status window opens messages that look something like the following:

```
Fatal error parsing the DTD - File: bad.dtd, line: 6, column: 2  
Message text: Invalid document structure
```

The first line tells you the name of the DTD file, as well as the line and column number where the parser detected the error. Note that this is the position that the parser detected the error, not necessarily where the error is. For example, if you are missing

the end of a comment in the DTD, the parser indicates where it first detected the error, not where you meant for the comment to end.

The second line displays the error message that was returned from the parser. The words "Message text:" are not part of the parser error message, but are generated by the utility.

In some cases, you can see the file name XMLBuffer. This means that the error was detected in the buffer of data that the utility generated. This can be caused by something such as a syntactically bad root element name, or an unreadable DTD file. To see the buffer of XML data that was created (XMLBuffer), you can look at the trace file that the utility creates. This file is named XMLTRC, and is located in the same directory as the DTD.

Note: The XMLTRC file can contain messages that start with "Ignoring validation error in...". These are a result of using an empty root element and do not indicate a problem.

Warning messages

There are also other warning messages that can be generated by the utility, even if the DTD is parsed without errors. These are issued when the utility finds something about the element structure that might indicate a problem. Warning messages are generated for the following conditions:

- If an element is declared as a descendent of itself, the following message is displayed:

Warning : Element "name" declared as a descendant of itself in the DTD file The nested instance of element "name" is ignored

An example of this is if element A has child elements B and C. However, element B has child A as one of its child elements. In this case, the utility ignores the nested occurrence of A, and omits it from the list of children for element B. A does not appear as a child of B in the EDI standard that is generated, and errors can occur if XML data is received that contains A as a child element of B. If incoming data does not use the nested occurrence of the element, and you do not need to map data to it, then this warning does not cause a problem.

- If the parser indicates that the root element has content model "Any", the following message is displayed:

Warning : Root element "name" has content model "Any" Element "name" might not be declared in the DTD file

There are two cases where the parser indicates that the root element has content model "Any". One case is that the DTD actually declares the element to be type "Any";. Although this is legal, this type of DTD is not very useful for mapping, because it does not define any structure for the DTD. The other case that can cause this is that the root element specified in the input field is not declared in the DTD file. (Note: XML element names are case sensitive.) In this case, the parser assumes that it is type "Any" . If you see this warning, verify that the root element name is correct for the DTD (including any case sensitivity). If not, correct it and retry. If the root element name is declared as type "Any", then you are not able to do much mapping for this DTD.

Utility JCL

The XML4C library (such as EDI.XML.SIXLMOD1) from the XML Toolkit must be included in the Utility JCL. For example:

```
//STEPLIB DD DSN=EDI.V3R3M0.SEDILMD1,DISP=SHR
// DD DSN=DB93.DSNEXIT,DISP=SHR
// DD DSN=DB93.DSNLOAD,DISP=SHR
// DD DSN=EDI.SNA131.LOADLIB,DISP=SHR
// DD DSN=EDI.XML.SIXLMOD1,DISP=SHR
```

The following files can be allocated for XML processing:

- XMLWORK—Required.
This is a work file or temporary file used for XML inbound and outbound processing. The file allocation for this is similar to FFSWORK. The WebSphere Data Interchange Utility and XML processor opens this file for OUTPUT and INPUT and it is always empty before the job begins. Use the JCL DISP=OLD to clear this file. It is a prime candidate for a virtual I/O data set.
- XMLERR—Required.
Error and status messages from the XML processor are written to this file. The record length is long enough to hold any messages from the XML processor (typically LRECL=255 is adequate), or some messages can be truncated. If you do not allocate this file in your JCL, then you are not able to see these messages. The file allocation for this is similar to PRTFILE. The XML processor opens the file for OUTPUT. It normally starts writing from the beginning of this file for each new PERFORM command in your job step. Use the JCL DISP options to control whether the file is cleared or appended to. If you are executing multiple PERFORM commands, you want to use DISP=MOD.
- XMLTRC—Optional.
If allocated, it contains XML processing trace messages. The file allocation for this is similar to EDITRACE. The XML processor opens the file for OUTPUT. It normally starts writing from the beginning of this file for each transaction on outbound, and each XML input file on inbound. Use the JCL DISP options to control whether the file is cleared or appended to.
- XMLEXCP—Required.
XML data that is in error is written to this file. If this file is not allocated in the JCL, the XML data that is in error is discarded. The file allocation for this is similar to FFSEXCP. The XML processor opens the file for OUTPUT when processing the first transaction, and starts writing from the beginning of the file. It then opens the file for extend for each subsequent transaction in the PERFORM command. Use the JCL DISP options to control whether the file is cleared or appended to during the first use. The file is allocated the same as your XML input file for inbound processing. If you are executing multiple PERFORM commands, you want to use DISP=MOD.

These files are used in a similar fashion as the related allocation files. The recommendation is to allocate these files the same as the related files.

XML Processor Messages and Codes

The XML processor generates the following error and status messages:

All messages are in the form: *XPnnnnS*, where:

nnnn is replaced by the message number

S tells the severity of the message. *S* can have the following values:

- I** means that the message is informational only, and is used to give status or to give additional information about other messages
- W** means that the message describes a warning. The current document is still considered valid, but an unusual condition was found that results in unexpected output.
- E** means that the message describes an error. The current document is considered invalid.
- T** means that the message describes a severe error that prevents any further processing by the XML processor.

Messages resulting from the XML processor can be found in the XMLERR file. XML data that is in error is written to the XMLEXCP file.

See the DTD Conversion Utility help for a complete list of error messages.

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, N.Y. 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
Department DD40
P.O. Box 30021
Tampa, Florida 33630-3021 USA

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, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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:

IBM
the IBM logo
AIX
CICS
IBMLink
IMS
WebSphere MQSeries
MVS
OS/390
WebSphere
z/OS

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

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

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

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.



Glossary of terms and abbreviations

This glossary defines WebSphere Data Interchange terms and abbreviations used in this book. If you do not find the term you are looking for, see the index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute. Copies may be ordered from the American National Standards Institute, 11 West 42 Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

A

AAR. Association of American Railroads. Represents the railroad industry in areas such as standards, public relations, and advertising.

acknowledgement. See *functional acknowledgement*, *network acknowledgement*.

ADF. See *data format*.

ANSI. American National Standards Institute.

ANSI ASC X12. ANSI Accredited Standards Committee X12, which develops and maintains generic standards for business transactions for EDI.

application. A program that processes business information. An application that requests services from WebSphere Data Interchange is an enabled application.

application data. The actual data in an application data file.

application data format. See *data format*.

application default profile. Identifies business applications, such as purchasing and accounts receivable, to WebSphere Data Interchange and sets specific WebSphere Data Interchange processing defaults for an application.

B

base structure. The data structure that contains all the data structures and data fields that define the application data for a single transaction.

binary format (BIN). Representation of a decimal value in which each field must be 2 or 4 bytes long. The sign (+ or -) is in the far left bit of the field, and the number value is in the remaining bits of the field. Positive numbers have a 0 in the sign bit. Negative numbers have a 1 in the sign bit and are in twos complement form.

C

CICS. Customer Information Control System.

CD-ROM. Compact Disk-Read Only Memory; a storage medium for large amounts of data needed external to the personal computer.

client-server. A computing environment in which two or more machines work together to achieve a common task.

code list. A table, supplied by WebSphere Data Interchange or defined by the user, that contains all acceptable values for a single data field.

composite data element. In EDI standards, a group of related subelements, such as the elements that make up a name and address.

compound element. An item in the source or target document that contains child items. Examples are EDI segments and composite data elements, data format records and structures, and XML elements.

control number. Numbers (or masks used to create numbers) that are used to identify an interchange, group, or EDI transaction.

control string. An object compiled from a map, data format, and EDI standard transaction; it contains the instructions used by the translator to translate a document from one format to another.

Glossary

control structure. The beginning and ending segments (header and trailer) of standard enveloped transmissions.

Customer Information Control System (CICS).

An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs.

customize. To alter to suit the needs of a company, such as removing from an EDI standard the segments and data elements that the company does not use.

D

data dictionary. A file containing the definitions of all the data elements of an EDI standard.

data element. A single item of data in an EDI standard, such as a purchase order number. Corresponds to a data field in a data format.

data element delimiter. A character, such as an asterisk (*), that follows the segment identifier and separates each data element in a segment. See also *element separator* and *segment ID separator*.

data field. A single item of data in a data format, such as a purchase order number. Corresponds to a data element in an EDI standard.

data format. A description of the application data for a particular transaction. A data format is composed of loops, records, data structures, and fields.

data format dictionary. A file that contains data format components.

data format record. A group of logically related fields set up as a record in a data format.

data format structure. A group of related data fields in a data format, such as the fields making up the line item of an invoice. Corresponds to a composite data element in an EDI standard.

DataInterchange/MVS™. The IBM DataInterchange product used on the host; pieces include a TSO parameter entry mechanism and a

translator. The functionality available in this product is now available in WebSphere Data Interchange for z/OS.

DataInterchange/MVS-CICS. The CICS-based IBM DataInterchange product. The functionality available in this product is now available in WebSphere Data Interchange for z/OS.

data structure. A group of related data fields in a data format, such as the fields making up the line item of an invoice. Corresponds to a segment in a standard.

Data Transformation map. One of three supported map types. A Data Transformation map is a set of mapping instructions that describes how to translate data from a source document into a target document. Both the source and target documents can be one of several support document types.

DB2®. Database 2, an IBM relational database management system.

ddname. Data definition name.

decimal notation. The character that represents a decimal point in the data.

delimiter. A character that terminates a string of characters, such as the value contained in a data element.

DI Client. WebSphere Data Interchange Client; the Windows-based, client/server interface for WebSphere Data Interchange.

dictionary. See *data dictionary*.

document. A business document that is exchanged between two enterprises as part of a business process, such as a purchase order or invoice. A document within WebSphere Data Interchange is singular. For example, it cannot contain multiple purchase orders. A document can also be represented in any syntax. For example, an XML purchase order and an EDI purchase order are both documents.

Document Type Definition (DTD). A list of all components included in the XML document and

their relationship to each other. This defines the structure of an XML document.

domain. The data structure or group of data structures in a data format to and from which you should restrict the mapping of EDI repeating segments and loops.

DTD. See *Document Type Definition*.

E

EDI. Electronic data interchange.

EDIA. Electronic Data Interchange Association.

EDI administrator. The person responsible for setting up and maintaining WebSphere Data Interchange.

EDI message. See *message*.

EDI standard. The industry-supplied, national, or international formats to which information is converted, allowing different computer systems and applications to interchange information.

EDI transaction. A single business document, such as an invoice.

EDI transaction set. A group of logically related data that make up an electronic business document, such as an invoice or purchase order.

EDIFACT. Electronic Data Interchange for Administration Commerce and Transport. See UN/EDIFACT.

electronic data interchange (EDI). A method of transmitting business information over a network, between business associates who agree to follow approved national or industry standards in translating and exchanging information.

electronic transmission. The means by which information is transferred between parties, such as over a public network.

element. See *data element*.

element separator. A character that separates the data elements in a segment. See also *data element delimiter*.

encryption. The encoding and scrambling of data. Data is encrypted by the sender and decrypted by the receiver using a predetermined program and unique electronic key.

event. An occurrence that is important to a user's computer tasks, such as a software error, sending a transaction, or acknowledging a message.

Extensible Markup Language (XML). A standard metalanguage for defining markup languages that was derived from, and is a subset of SGML. It is used to represent structured documents and data.

F

field. See *data field*.

floating segment. A segment of an EDI standard that may exist in many positions relative to other segments.

forward translation table. A user-defined table that translates data values that differ between trading partners. For example, if a manufacturer and supplier have different part numbers for the same item, each company can use its own part number and have it converted to the other company's part number during translation. Forward translation tables translate local values to standard values.

functional acknowledgement. An electronic acknowledgement returned to the sender to indicate acceptance or rejection of EDI transactions.

functional group. One or more transaction sets of a similar type transmitted from the same location, enclosed by functional group header and trailer segments.

Glossary

G

global variable.. A variable that is shared among all instances of all documents within a translation session.

H

header. A control structure that indicates the start of an electronic transmission.

hierarchical loop. A technique for describing the relationship of data entities which are related in a parent/child manner, like a corporate organization chart. Used in mapping to group related data elements and segments such as trading partner address.

HL. *See hierarchical loop.*

I

IBM Global Network. The IBM communications network that provides products and services to IBM customers.

ICS. International Control Segments.

import. The process of taking WebSphere Data Interchange objects exported on another WebSphere Data Interchange system and incorporating them into the receiving system.

Information Exchange. A commerce engine of IBM Interchange Services for e-business that permits users to send and receive information electronically.

interchange. The exchange of information between trading partners.

J

JCL. Job Control Language.

K

key. In a profile member, the field that identifies the member. For example, the key for members of the trading partner profile is the trading partner nickname.

L

literal. In mapping, a value that is constant for each occurrence of the translation. If you provide the literal value during mapping, the translator does not have to refer repeatedly to the source to obtain the value.

local variable. A variable that is specific to the instance of the document in which it is being used.

log file. A file in which events are recorded.

logging. The recording of events in time sequence.

loop. A repeating group of related segments in a transaction set or a repeating group of related records and loops in a data format.

loop ID. A unique code identifying a loop and the number of times the group can be repeated.

loop repeat. A number indicating the maximum number of times a loop can be used in a transaction set.

M

mailbox. If you use a mail type protocol to exchange messages with your trading partners, you will have one or more registered mailboxes. The mailbox profile is used in WebSphere Data Interchange to define your mailboxes and any associated preferences.

map. A set of instructions that indicate to WebSphere Data Interchange how to translate data from one format to another.

map rule. An association between a Data Transformation map and a trading partner.

maximum use. A number indicating the maximum number of times a segment can be used in a transaction set or the maximum number of times that a data format loop or record can repeat.

message. A free-form, usually short, communication to a trading partner. In UN/EDIFACT standards, a group of logically related data that make up an electronic business document, such as an invoice. A message is equivalent to a document.

message log. The file in which WebSphere Data Interchange Client logs messages about errors that occur within the client.

multiple-occurrence mapping. A form of mapping in which all occurrences of a loop or repeating segment are mapped to the same repeating structure in the data format.

N

network acknowledgement. A response from the network indicating the status of an interchange envelope, such as sent or received.

network commands. The commands that you want WebSphere Data Interchange to pass to your network, defined in the network commands profile. In the host product, this file is named NETOP.

O

ODETTE. Organization for Data Exchange through Teletransmission in Europe.

P

parse. To break down into component parts.

path qualified mapping. A form of mapping in which all occurrences of a repeating compound or simple data element are mapped to a repeating compound or simple data element in another document.

PDS. Partitioned data set.

PDS members. Groups of related information stored in partitioned data sets.

profile. Descriptive information about trading partners, network connections, and so on. Each profile can contain one or more objects or members. For example, the trading partner profile contains members for your trading partners (one member for trading partner address).

program directory. A document shipped with each release of a product that describes the detailed content of the product.

Q

qualifier. A data element which gives a generic segment or data element a specific meaning. Qualifiers are used in mapping single or multiple occurrences.

R

Receive map. One of three supported map types. A Receive map is a set of mapping instructions that describe how to translate an EDI standard transaction into a proprietary application data document.

receive usage. An association between a Receive map and a trading partner.

record. A logical grouping of related data structures and fields.

release character. The character that indicates that a separator or delimiter is to be used as text data instead of as a separator or delimiter. The release character must immediately precede the delimiter.

repository data. A group of data definitions, formats, and rules/usages, that WebSphere Data Interchange uses to process your data.

requestor. See *mailbox*.

reverse translation table. A user-defined table that translates data values that differ between trading partners. For example, if a manufacturer and supplier have different part numbers for the

Glossary

same item, each company can use its own part number and have it converted to the other company's part number during translation. Reverse translation tables translate standard values to local values.

rule. See *map rule*.

runtime data. Data used by the WebSphere Data Interchange translator, such as control strings, code lists, translation tables and profiles.

S

security administrator. The person who controls access to business data and program functions.

segment. A group of related data elements. A segment is a single line in a transaction set, beginning with a function identifier and ending with a segment terminator delimiter. The data elements in the segment are separated by data element delimiters.

segment directory. A file containing the format of all segments in an EDI standard.

segment identifier. A unique identifier at the beginning of each segment consisting of two or three alphanumeric characters.

segment ID separator. The character that separates the segment identifier from the data elements in the segment.

segment terminator. The character that marks the end of a segment.

Send map. One of three supported map types. A Send map is a set of mapping instructions that describe how to translate a proprietary application data document into an EDI standard transaction.

send usage. An association between a Send map and a trading partner.

simple element. An item in the source or target document that does not contain child items, only data. Examples are EDI data elements, data format fields, XML attributes, and PCDATA values.

single-occurrence mapping. A form of mapping in which each occurrence of a loop or repeating compound or simple data element in a document is mapped to a different compound or simple data element in another document.

source document definition. A description of the document layout that will be used to identify the format of the input document for a translation.

special literal. The send and receive Mapping Data Element Editors include the Literal or Mapping Command field. Literals are constant values you enter in this field, such as 123. Special literals are values you enter in this field that begin with an ampersand (&) and are command to WebSphere Data Interchange, rather than constant values. For example, to use today's date, you enter &DATE.

standards. See *EDI standard*.

structure. See *data structure* or *data format structure*.

subelement. In UN/EDIFACT standards, a data element that is part of a composite data element. For example, a data element and its qualifier are subelements of a composite data element.

subelement separator. A character that separates the subelements in a composite data element.

T

tag. In UN/EDIFACT standards, the segment identifier. In export/import, a code identifies each field in the export record. Such export/import files are known as "tagged" files.

target document definition. A description of the document layout that will be used to create an output document from a translation.

TD queue. See *transient data queue*.

TDCC. Transportation Data Coordinating Committee.

TDQ. Transient data queue.

temporary storage queue (TS). Storage locations reserved for immediate results in CICS. They are deleted after the task that created them is complete and they are no longer necessary.

TPT. Trading partner transaction. See *map*.

trading partner profile. The profile that defines your trading partners, including information about network account numbers, user IDs, who pays for network charges, etc.

trading partners. Business associates, such as a manufacturer and a supplier, who agree to exchange information using electronic data interchange.

trading partner transaction. See *map*.

trailer. A control structure that indicates the end of an electronic transmission.

transaction. A single business document, such as an invoice. See also *EDI transaction*.

transaction set. A group of standard data segments, in a predefined sequence, needed to provide all of the data required to define a complete transaction, such as an invoice or purchase order. See also *EDI transaction set*.

Transaction Store. The file that contains the results of translations and a history of translation activity.

transform. The process of converting a document from one format to another.

transient data queue (TD). A sequential data set used by the Folder Application Facility in CICS to log system messages.

translation. The process of converting a document from one format to another.

translation table. A user-defined table that translates data values that differ between trading partners. For example, if a manufacturer and supplier have different part numbers for the same item, each company can use its own part number and have it converted to the other company's part number during translation.

TSQ. See *temporary storage queue*.

U

UCS. Uniform Communication Standard.

unary operator. An operator that changes the sign of a numeric value.

UN/EDIFACT. United Nations Electronic Data Interchange for Administration Commerce and Transport.

Uniform Communication Standard (UCS). The EDI standard used in the grocery industry.

UN/TDI. United Nations Trade Data Interchange.

Usage. An association between a send or Receive map and a trading partner.

V

validation table. A table, supplied by WebSphere Data Interchange or defined by the user, which contains all acceptable values for a single data field.

variable. The entity in which a value may be stored based on data received; as opposed to a constant value.

W

WebSphere Data Interchange. A generic term for the WebSphere Data Interchange products, WebSphere Data Interchange for z/OS and WebSphere Data Interchange for Multiplatforms. WebSphere Data Interchange is a translator of data from one document format to another; the pieces of this product include a TSO parameter entry mechanism, a CICS parameter entry mechanism, a Windows-based parameter entry mechanism (WebSphere Data Interchange Client), and a translator.

WebSphere Data Interchange Client. A Windows-based product for entry of parameters needed by the WebSphere Data Interchange translator.

Glossary

WebSphere MQ. An IBM product that is used to implement messaging and queueing of data groups. Earlier releases of this product were known as MQSeries®.

WebSphere MQ queue profile. Represents a relationship between a logical name and a physical WebSphere MQ queue name.

WINS. Warehouse Information Network Standard.

Windows®. Microsoft's graphical operating system under which WebSphere Data Interchange Client runs.

X

X12. A common EDI standard approved by the American National Standards Institute.

XML. See *Extensible Markup Language*.

Bibliography

This section describes the documentation available for the WebSphere Data Interchange product.

WebSphere Data Interchange publications

The WebSphere Data Interchange V3.3 publications are:

- *WebSphere Data Interchange for MultiPlatforms Quick Start Guide* CF0YREN
- *WebSphere Data Interchange for MultiPlatforms Administration and Security Guide* SC34-6214-01
- *WebSphere Data Interchange for MultiPlatforms Messages and Codes Guide* SC34-6216-01
- *WebSphere Data Interchange for MultiPlatforms User's Guide* SC34-6215-01
- *WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide* SC34-6217-01
- *WebSphere Data Interchange for MultiPlatforms Mapping Guide* SC23-5874-00
- *WebSphere Data Interchange for MultiPlatforms Utility Commands and File Formats Reference Guide* SC23-5873-00
- *WebSphere Data Interchange for z/OS V3.3 Program Directory* G110-2561-01
- *WebSphere Data Interchange for z/OS V3.3 Installation Guide* SC34-6269-01
- *WebSphere Data Interchange for z/OS V3.3 License File* GC34-6270-02

Softcopy books

All the WebSphere Data Interchange books are available in softcopy format.

Portable Document Format (PDF)

The library is supplied as stand-alone PDFs in US English in the DOC directory on the product CD. The contents of the DOC directory can be viewed without installing the product.

PDF files can be viewed and printed using the Adobe Acrobat Reader. You will need Adobe Acrobat Reader with Search Version 4.05 on Windows NT, or Adobe Acrobat Reader with Search Version 4.5 on UNIX® systems.

If you need to obtain the Adobe Acrobat Reader, or would like up-to-date information about the platforms on which the Acrobat Reader is supported, visit the Adobe Systems Inc. Web site at:

<http://www.adobe.com/>

If you cut and paste examples of commands from PDF files to a command line for execution, you must check that the content is correct before you press Enter. Some characters might be corrupted by local system and font settings.

WebSphere Data Interchange information available on the Internet

The WebSphere Data Interchange product Web site is at:

<http://www.ibm.com/websphere/datainterchange/>

By following links from this Web site you can:

- Obtain latest information about the WebSphere Data Interchange products.
- Access the WebSphere Data Interchange books in PDF format.

Bibliography

Index

Numerics

841 transaction set 285

A

accumulators

- actions 242
- adding to maps 242
- types 242
- types of 242
- using 241

application control key, setting 209

application data

- components
 - determining 28
 - fields 29
 - loops 30
 - records 29
 - structures 29
- obtaining 27
- raw data records 27
- structuring 27

assignment 147

B

BIN segment ID 285

binary segment receive processing 293

C

Char function 175

CloseOccurrence command 149

code list 73

commands

- CloseOccurrence 149
- conditional 147
 - Else 147
 - Elseif 147
 - EndIf 147
 - If 147
- Create 150
- Default 162
- Error 150
- ErrorContext 151
- FAError 151
- FAErrorPath 155
- ForEach 155
- HLLLevel 158
- MapCall 158
- MapChain 159
- MapFrom 159
- MapSwitch 160
- MapTo 161
- overview 149

commands (*continued*)

- Qualify 162
- SetElementAttribute 163
- SetNamespace 172
- SetNoNSSchemaLocation 173
- SetProperty 174
- SetSchemaLocation 173

comments 143

Concat function 176

conditional commands 147

control and data format 28

control strings

- compiling 116
- viewing compiled 116

Create command 150

Created function 176

D

data elements

- advanced mapping capabilities 203
- creating 81
- editor 202
- parts of 72
- qualifying 114

data format dictionary editor

- creating 35
- importing a COBOL copybook 36

data format editors

- accessing 34
- dictionary editor 35
- field editor 49
- loop editor 43
- navigating
 - data formats editor paths 51
 - dictionary editor paths 50
 - field editor paths 53
 - loop editor paths 52
 - overview 50
 - record editor paths 52
 - structure editor paths 53
- record editor 45
- record ID information editor 38
- structure editor 47
- using 33, 40

data format worksheet 31

data formats 25

- creating 26, 40

data types for

- A (alphabetic) 54
- AC (application control) 54
- AN (alphanumeric) 55
- Bn (binary–unsigned) 55

data formats *(continued)*

data types for *(continued)*

- BN (binary–unsigned) 55
- CH (character) 55
- DT (date) 56
- FN (file name) 56
- Hn (hexadecimal) 56
- HX (hexadecimal) 56
- ID (identifier) 56
- ln (integer–signed) 57
- IT (integer–signed) 57
- IV (integer–signed) 57
- Ln (decimal–leading sign) 57
- N (numeric) 58
- Nn (numeric) 58
- PD (packed decimal) 58
- Pn (packed decimal) 58
- PW (password) 59
- R (real) 59
- Rn (real) 59
- TM (time) 59
- understanding 54
- ZD (zoned decimal) 59
- Zn (zoned decimal) 60

reusing components 53

Data Transformation

Functional Acknowledgement map editor

- starting 136
- using 137

map editor

- using 121

validation map editor

- starting 131
- using 132

Data Transformation maps

- commands 141
- commands, conditional 147
- creating 122, 132, 137
- data types 144
- details tab 95
- editing 97
- expressions 145
- forward reference 144
- functions 141
- keywords 143
- map rules
 - applying 125
 - viewing 126
- operators
 - arithmetic 146
 - comparison 146
 - logical 145
 - modulus 146
 - order 147
 - unary 147

Data Transformation maps *(continued)*

path 143

qualification

- editing an element qualification 108
- specifying 103
- types of 103

qualifying

- repeating simple and compound elements 104
- repeating simple and compound elements
 - multi-occurrence 106
- repeating simple and compound elements
 - occurrence 105
- repeating simple and compound elements:by
 - expression 108
- repeating simple and compound elements:by
 - value 107

reverse reference 144

data types

- conditions 211
- for data formats 54
- understanding 54
- with mapping commands 144

Date function 177

DateCnv function 177

Default command 162

dictionary 72

document type definition (DTD) 61

DTD Conversion Utility

- converting 321
- diagnosing errors 341
- DTD alias 329
- DTD, processing external 327
- encoding considerations 336
- example 328
- output 325
- PERFORM commands 334
- prolog, overriding the default 337
- translation
 - inbound 329
 - outbound 331
- updating 327
- XML dictionary 326

DTD editor 64

E

EDI standard dictionary

- creating a 74

EDI standards 71

EDI standards editors

- accessing 74
- code list editor 84
- data element editor 81
- dictionary editor 74
- envelope
 - control string 87

- EDI standards editors *(continued)*
 - envelope *(continued)*
 - editor 84
 - fields descriptions 85
 - segment editor 79
 - transaction editor 76
 - transaction, creating 76
 - using 73
- editor, map 93
- EFI segment 287
- electronic format identification segment 287
- elements, repeating 104
- Else conditional command 147
- Elseif conditional command 147
- encoding
 - application data
 - incoming 314
 - outgoing 314
 - EDI data
 - incoming 314
 - outgoing 315
 - XML data
 - incoming 315
 - outgoing 315
- EndIf conditional command 147
- envelope standard
 - compiling 87
 - editor 85
- envelopes 71
- Error command 150
- ErrorContext command 151
- examples
 - international tasks 319
 - literal keywords 270
 - named variables 270
 - PC executable files 290
 - sending to limited systems 291
 - variable length PC files 291
- Exit function 177
- expressions 145, 258
- extensible markup language (XML)
 - See XML

F

- FAError command 151
- FAErrorPath command 155
- fields
 - advanced mapping 203
 - application 291, 296
 - creating 49
 - description 29
 - editor path 53
 - MQ profile 316
 - service segment
 - generic 280

- fields *(continued)*
 - service segment *(continued)*
 - Receive mapping 279
 - Send mapping 278
 - using 277
- Find function 178
- fixed-to-fixed maps 240
- ForEach command 155
- format specifications 289
- Found function 178
- functions
 - Char 175
 - Concat 176
 - Created 176
 - Date 177
 - DateCnv 177
 - Exit 177
 - Find 178
 - Found 178
 - GetProperty 179
 - HexDecode 180
 - HexEncode 179
 - IsEmpty 180
 - Left 181
 - Length 181
 - Lower 182
 - Number 182
 - NumFormat 182
 - Occurrence 183
 - Overlay 184
 - overview 175
 - Right 185
 - Round 185
 - StrComp 186
 - StrCompl 186
 - StrCompN 187
 - StrCompNI 187
 - SubString 188
 - Time 189
 - Translate 189
 - TrimLeft 190
 - TrimRight 191
 - Truncate 191
 - Upper 192
 - Validate 192

G

- generic receive usages 239
- generic send usages 238
- GetProperty function 179
- global accumulator 242

H

- handling international characters
 - automatic detection 314

- handling international characters *(continued)*
 - byte-order mark support 317
 - considerations for import and export 316
 - encoding
 - incoming application data 314
 - incoming EDI data 314
 - incoming XML data 315
 - outgoing application data 314
 - outgoing EDI data 315
 - outgoing XML data 315
 - examples 319
 - MQ profile fields 316
 - overview 313
 - PERFORM keywords 316
 - processing 314
- HexDecode function 180
- HexEncode function 179
- hierarchical loops
 - examples 297
 - in Data Transformation maps
 - adding levels 306
 - comparing 307
 - creating 300
 - defining 300
 - overview 300
 - qualifying 301
 - source based mapping 302
 - special mapping 302
 - target based mapping 303
 - in send or Receive maps
 - overview 310
 - segment literal keyword 311
 - mapping 123
 - mapping HL segment 310
 - overview 297
 - preparing 299
 - segments 298
 - specifying levels 297
- HLLLevel command 158

I

- If conditional command 147
- information on the Internet
 - WebSphere Data Interchange 357
 - WebSphere Data Interchange libraries 357
- intellectual property 345
- IsEmpty function 180

K

- keywords 143

L

- Left function 181
- Length function 181
- license, patents 345

- licensing
 - address 345
- literal keywords
 - conditional processing 245
 - examples 270
 - keywords 246
 - &ACFIELD 246
 - &ASSERT 247
 - &DATE 247
 - &DEFERRED 248
 - &E 248
 - &ERR 249
 - &FORCE 249
 - &FORMAT 249
 - &IF 250
 - &IFDATA 250
 - &IFNODATA 250
 - &IFNOVAR 251
 - &LOOPBREAK 251
 - &LOOPCHECK 251
 - &LSAVE 252
 - &LSET 253
 - &LSID 253
 - &SAMEAS 253
 - &SAVE 254
 - &SET 255
 - &THANDLE 255
 - &TIME 255
 - &TPID 255
 - &TPNICKN 255
 - &USE 256
 - &ZEROSIG 256
 - mapping techniques 268

literals

- accumulator 245
 - adding to a map 211
 - and data types 211
 - conditional processing 245
 - control data 277
 - format 245
 - keywords
 - See literal keywords
 - overview 142
 - using 211, 243
 - using for Receive mapping 244
- loops
 - creating 43
 - hierarchical
 - See hierarchical loops
 - overview 30
 - qualifying 110
- Lower function 182

M

- map editor
 - editing procedure 95
 - panes
 - Data Transformation maps 93
 - Functional Acknowledgement maps 94
 - Receive maps 94
 - Send maps 94
 - using 99
 - validation maps 94
 - starting 95
 - symbols 98
 - using 93
- map rules
 - applying 125
 - viewing 126
- map variables 141
- MapCall command 158
- MapChain command 159
- MapFrom command 159
- mapping
 - advanced techniques 100
 - choosing the right map 102
 - generic service segment fields (receive) 280
 - getting started 100
 - service segment fields (send) 278
 - specific service segment fields (receive) 279
- mapping commands
 - Data Transformation maps
 - data types 144
 - overview 141
 - send and Receive maps
 - adding to a map 211
 - using 211
- mapping data element editor
 - application control key 209
 - applying advanced mapping capabilities 203
 - applying advanced mapping capabilities with 99
 - date conversion operator 206
 - options, special handling 204
 - overview 202
 - repeat button 209
 - special handling button 203
 - user exit profile 208
- maps
 - choosing 102
 - creating 93, 100
 - Data Transformation 121
 - fixed-to-fixed 240
 - functional acknowledgement 135
 - migrating 117
 - objects 89
 - receive 201
 - send 201
 - task list 100

- maps (*continued*)
 - types 7
 - validation 131
- MapSwitch command 160
- MapTo command 161
- migrating a map 117
- minimal trading partners 125, 215

N

- Number function 182
- NumFormat function 182

O

- Occurrence function 183
- operators
 - arithmetic 146
 - comparison 146
 - date conversion 206
 - logical 145
 - modulus 146
 - unary 147
- order of precedence 147
- Overlay function 184

P

- patents 345
- path, specifying 143
- PDF (Portable Document Format) 357
- Portable Document Format (PDF) 357
- properties
 - EDI envelope standard generic 195
 - EDI envelope standard specific 196
 - source document 193
 - target document 194
- publications
 - WebSphere Data Interchange 357

Q

- qualification
 - editing a data element 115
 - editing a loop 113
 - editing a segment 113
 - editing an element qualification 108
 - specifying 103, 110
 - types of 103, 110
- Qualify command 162
- qualifying
 - composite data element
 - by path 109
 - data elements
 - by value 114
 - editing 115
 - overview 114
 - loops
 - by occurrence 111
 - by path 111

- qualifying (*continued*)
 - loops (*continued*)
 - by value 112
 - overview 110
 - repeating data element
 - by occurrence 109
 - by path 109
 - repeating simple and compound elements
 - by expression 108
 - by multi-occurrence 106
 - by occurrence 105
 - by value 107
 - overview 104
 - segments
 - by occurrence 111
 - by path 111
 - by value 112
 - overview 110

R

- Receive maps
 - advanced techniques 100
 - creating 201
 - qualification
 - editing a data element 115
 - editing a loop 113
 - editing a segment 113
 - specifying 110
 - types of 110
 - qualifying
 - data elements 114
 - data elements: by value 114
 - loops 110
 - loops: by occurrence 111
 - loops: by path 111
 - loops: by value 112
 - segments 110
 - segments: by occurrence 111
 - segments: by path 111
 - segments: by value 112
 - qualifying a composite data element
 - by occurrence 109
 - by path 109
 - qualifying a repeating data element
 - by occurrence 109
 - by path 109
 - translation tables 212
 - creating 214
- receive processing, binary segment 293
- records 29
 - creating 45
 - definition 29
 - editor path 52
 - ID information 38

- references
 - forward 144
 - reverse 144
- repeating elements 104
- Right function 185
- Round function 185

S

- segment ID
 - BIN 285
 - BIN length 285
- segments
 - BDS 286
 - BIN 285
 - binary 288
 - creating a 79
 - creation for Send maps 244
 - definition 72
 - EFI 287
 - mapping for Receive maps 279
 - mapping for Send maps 278
 - mapping generic for Receive maps 280
 - qualifying
 - by occurrence 111
 - by path 111
 - by value 112
 - editing 113
 - overview 110
 - receive processing 293
 - segment editor 79
 - send processing 288
 - service 277
- sending
 - PC executable files 290
 - PC variable length files 291
 - to systems with file limitations 291
- service segment fields 277
- set 841 285
- SetElementAttribute command 163
- SetNamespace command 172
- SetNoNSSchemaLocation command 173
- SetProperty command 174
- SetSchemaLocation command 173
- softcopy books 357
- special operators 206
- StrComp function 186
- StrCompI function 186
- StrCompN function 187
- StrCompNI function 187
- structures
 - creating 47
 - definition 29
- SubString function 188

T

- task list, mapping 100
- Time function 189
- trading partners
 - copying usages of 238
 - creating usages for 216, 225
 - editing usages of 238
 - minimal concept 215
 - specifying usages of 215
 - viewing usages of 126
- transaction accumulator 242
- transaction sets 72
- transactions
 - BIN segment length 285
 - electronic format identification (EFI) 287
 - overview 72
 - parts of
 - code list 73
 - data elements 72
 - dictionary 72
 - segments 72
 - transaction sets 72
- Translate function 189
- translation tables
 - creating 214
 - data differences 212
 - description 212
 - forward 213
 - reverse 213
- TrimLeft function 190
- TrimRight function 191
- Truncate function 191

U

- Upper function 192
- usages
 - copying 238
 - creating
 - Receive map 225
 - Send map 216
 - editing 238
 - generic
 - receive 239
 - send 238
 - specifying 215
 - viewing 216
- user exit profile 208

V

- Validate function 192
- validation 282
- validation map
 - creating 132
 - editor 131
 - overview 131

- validation map (*continued*)
 - using 132
- variables
 - Data Transformation
 - local 141
 - naming 142
 - global 142
 - creating 90
 - overview 13
 - loop 141
 - Receive map
 - examples 270
 - local 256
 - variables 266
 - Send map
 - examples 270
 - local 256
 - variables 266
 - special 142

W

- WebSphere Data Interchange on the Internet 357
- WebSphere Data Interchange publications 357
- worksheet, data format 31

X

- XML
 - dictionary 62
 - document type definition (DTD) 61
 - DTD, importing 63
 - editor 62
 - Namespace 65
 - overview 61
 - processing 66
 - receiving data using VAN 330
- XML dictionary
 - creating 62
- XML dictionary editor 62
- XML editors 62



Printed in USA

SC23-5874-00

