
Version 6.2.0

MW Translator Workbench User's Guide and Reference



Document History

Part Number	Product Name	Date
MWT600-610	MW Translator Workbench User's Guide and Reference	11/2011
MWT610-610	Same as above	11/2012
MWT620-610	Same as above	06/2018

Copyright

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Chef, Chef (and design), Chef Infra, Code Can (and design), Compliance at Velocity, Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

This document was published on Tuesday, May 09, 2023 at 17:13.

MW Translator Workbench User's Guide and Reference

Contents

Basic Information	1
Typographical Conventions.....	1
Purpose of the MW Translator Workbench.....	2
Audience for MW Translator.....	2
What You Can Find Here	2
Basic Features	3
Getting Started	5
Technical Support.....	5
Related Information	6
What Features are Supported.....	7
System Requirements and Recommendations.....	8
Utilities	8
Working Examples.....	9
Additional Standards.....	9
MW Translator Database Environments.....	10
Installing Additional Database Environments.....	10
Converting Previous Versions of the MW Translator Configuration Databases.....	10
Planning and Configuration	11
Overview	11
Determining Your Trading Requirements	11
How to Convert Your Trading Requirements to Configurations.....	12

Basic Questions to Determine Trading Requirements 13

Using a Template Approach for New Configurations 14

Integrating Test and Production Processes 15

Configuration Task Tree 16

Determining Processing Control **17**

Overview (Determining Processing Control)..... 17

Identifying the Incoming Standard..... 17

 Defining Inbound Locations 17

 Defining Matching Criteria..... 18

Specifying a Trade Agreement Profile..... 20

 Criteria to Define Senders or Recipients..... 21

 Understanding Trade Agreement Profiles 22

 Deciding What Type of Trade Relationship You Need 23

 Processing Step - Looking for Defined Partner IDs 24

 Processing Step - Finding Partner Profiles for the Sender and Recipient..... 24

 Processing Step - Finding a Partner Relationship Definition 24

 Processing Step - Repeating Preceding Steps for Remaining Wrapper Levels 25

Specifying Routing Locations 25

 Locations for Output..... 25

 Processing Step - Finding Routing Locations for Translation..... 26

 Processing Step - Finding Routing Locations for Validation 26

 Locations for Backward Acknowledgments..... 27

 Processing Step - Finding Routing Locations for Backward Acknowledgments 27

Quick Configuration Tour and Test 29

- Overview29
 - Processing Flow30
 - Processing Options31
- Validate the Wrapper and Route the Input31
- Validate Wrapper and Contents and Route the Input33
- Translate.....36
- Reviewing Sample Configurations.....38
 - Checking Your Environment38
 - Checking for the Example Standard.....38
 - Configurations for EXAMPLE Test40
- Testing the EXAMPLE Translation40
 - Running the Example Test41
 - Viewing the Reports and Output43

Understanding Processing Flow 45

- Overview45
- How the TRM Processes Documents46
 - Step 1. Identify the Incoming Standard47
 - Step 2. Execute a Pre-processing User Exit, as required.49
 - Step 3. Parse and Validate One Level of the Incoming Wrapper.....50
 - Step 4. Look for Defined Partner IDs for the Sender and Recipient at this Level.....53
 - Step 5. Find Partner Profiles for the Sender and Recipient Based on Partner IDs55
 - Step 6. Find a Partner Relationship Definition, If One Exists57
 - Step 7. Repeat Steps 3-6 for Remaining Wrapper Levels 2 through 4, if They Exist57

Step 8. Find the Acknowledgment Profiles, if They Exist.....	57
Step 9. Find a Trade Agreement Profile.....	61
Step 10. Parse the Incoming Document(s) (Contents Validation and Translation)	68
Step 11. Generate Outbound Documents	70
Step 12. Repeat Steps 9-11 for Remaining Documents.....	74
Step 13. Determine IDs for Outbound Partners	74
Step 14. Generate (Validation Only or Translation)	86
Step 15. Determine Routing (Validation).....	91
Step 16. Determine Routing (Translation).....	91
Step 17. Repeat Steps 14 And 16 for Each Defined Output (Translation)	97
Step 18. Create Backward Acknowledgments, As Required.....	99
Step 19. Determine Routing for Backward Acknowledgments	103

Map Development Guidelines 107

Use of EDI Standards..... 107

Review of Proprietary Formats 108

Definition of Proprietary Formats..... 110

Standard Identification 110

Wrapper Map..... 111

Document Map 111

Partners and Routing 112

Software Release..... 112

Defining Standards 115

Overview 115

Defining Standards..... 115

Task 1. Creating a Proprietary Standard 116

 Task Tree to Define Standards..... 117

Task 1.1. Specifying Your Standard and Version 117

 Task Tree to Define a Standard..... 118

 Task 1.1.1. Defining Your Standard and Version 118

 Task 1.1.2. Defining the Standard Types 119

 Task 1.1.3. Defining Standard Version Properties 120

Task 1.2. Defining the Content..... 120

 Task Tree to Define the Content..... 121

 Task 1.2.1. Defining the Document 122

 Task 1.2.2. Specifying Document Properties 123

 Task 1.2.3. Defining the Segments..... 124

Task 1.2.4. Specifying the Segment Properties.....	125
Task 1.2.5. Defining the Elements	126
Task 1.2.6. Specifying Element Properties.....	127
Task 1.2.7. Generating the Text File for the Document.....	127
Task 1.3. Defining the Envelope (Wrappers).....	127
Task Tree to Define the Envelope.....	128
Task 1.3.1. Defining the Wrapper	129
Task 1.3.2. Specifying Wrapper Properties.....	130
Task 1.3.3. Defining the Segments.....	136
Task 1.3.4. Specifying the Segment Properties.....	139
Task 1.3.5. Defining the Elements	140
Task 1.3.6. Specifying Element Properties.....	141
Task 1.3.7. Generating the Text File for the Wrapper	142
Task 1.4. Printing Document and Wrapper Reports	143
Special Cases. Defining Standards.....	145
Configuring Null Wrappers	146
Configuring Variable-length Segments in Fixed-length Standards	153
Creating Maps	155
Overview	155
Types of Mapping	155
Sources for Output.....	158
Understanding Visual and Edibasic Mapping Techniques	159
Creating a Map to Learn Mapping Techniques.....	159
Using Drag-and-Drop	160

Visual Mapping Techniques	161
Visual Mapping Default Behavior	163
Using Mapping Tools	164
Edibasic Mapping Techniques.....	165
Deleting Your Learning Map	166
Task 2. Creating Proprietary Maps.....	167
Overview of Creating Proprietary Maps	167
Creating Maps Task Tree	167
Task 2.1. Creating a Wrapper Map	168
Task Tree to Create a Wrapper Map	168
Task 2.1.1. Specifying the Map Name, Description, and Type for the Wrapper	169
Task 2.1.2. Selecting the Source Definition	169
Task 2.1.3. Selecting the Destination Document.....	170
Task 2.1.4. Entering Mapping Instructions for the Wrapper	170
Task 2.1.5. Generating the Wrapper Map	171
Task 2.2. Creating a Document Map.....	171
Task Tree to Create a Document Map.....	171
Task 2.2.1. Specifying the Map Name, Description, and Type for the Document.....	172
Task 2.2.2. Selecting the Source Definition	173
Task 2.2.3. Selecting the Destination Definition.....	174
Task 2.2.4. Entering Mapping Instructions for the Document	175
Task 2.2.5. Generating the Document Map.....	182

Task 2.3. Printing Map Reports	183
Advanced Mapping Techniques	185
To Display the Contents of the Data Element Store	185
Understanding the Context List in the Data Element Store	186
Level in DES Versus Level in Document Configuration	188
Understanding Levels in the Context List	189
Effect of Visual Mapping on Context List.....	190
Effect of Edibasic Commands on the Context List	200
Example of Context List Using Drag-and-Drop for Dissimilar Structures	202
Example of Context List Using Edibasic for Dissimilar Structures.....	204
Accessing Input Directly Using Edibasic without Using the Context List	212
Edibasic Solutions.....	212
Using Conditions to Create Output.....	212
Mapping Values from an Input Trailer Wrapper	213
Using Edibasic to Specify Output Locations.....	214
Defining Trade Agreements and Acknowledgments	219
Overview	219
Trade Agreement Profiles Process Inbound Data.....	220
Acknowledgment Profiles Generate Outbound Acknowledgments.....	221
Task 3. Defining Trade Agreement Profiles.....	222
Task 3.1. Specifying the Matching Fields	222
Task 3.2. Specifying the Options.....	223
Task 3.3. Specifying the Input Document	225
Task 3.4. Specifying the Outputs	226

Task 3.5. Defining the Trade Agreement Output Properties	227
Task 3.6. Generating the Trade Agreement Profile	228
Task 3.7. Printing a Trade Agreement Profile Report	229
Task 4. Defining Acknowledgment Profiles.....	230
Task 4.1. Specifying the Input Wrapper.....	230
Task 4.2. Specifying the Maps.....	231
Task 4.3. Specifying the Options.....	232
Task 4.4. Specifying the Summary Document and Security User Exit	233
Task 4.5. Generating the Acknowledgment Profile	233
Task 4.6. Printing an Acknowledgment Profile Report	234
Understanding Acknowledgments.....	235
Methods to Report Acknowledgment Information	235
Purpose of Acknowledgments	236
Levels of Acknowledgment Response.....	236
Acknowledgments Use Statuses.....	236
Special Cases for Trade Agreements.....	239
Configuring a Trade Agreement to Process an Acknowledgment without Mapping	239

Defining Partners 243

Overview of Defining Partners.....	243
Using Partner Explorer or the Partner Wizard	244
Partner Information	244
Partner Relationships.....	245
Alternate IDs and Locations for Outputs	246
Aliases	249

Using the Partner Wizard.....	250
Configuring Partner Information.....	251
Using Internal Fields for Partner Information.....	251
Associate Elements with Internal Fields	252
Defining Partner IDs for Partners.....	254
Configuring Partners to Control Processing.....	257
Specifying Alternative Output IDs or Routing.....	257
Using Trade Agreement Options	258
Using Aliases	263
Considering Additional Processing Controls	265
Restricting Processing to Defined Partner.....	266
Using Default Partner Definitions.....	266
Task 5. Creating Partner Definitions	267
Task 5.1. Creating a Partner Relationship.....	267
Task 5.1.1. Specifying Partner Relationship Name	267
Task 5.1.2. Specifying Partner IDs.....	269
Task 5.1.3. Specifying the Trade Agreement Profile.....	271
Task 5.1.4. Specifying Additional Information for the Outputs.....	273
Task 5.1.5. Specifying Acknowledgment Profiles.....	276
Task 5.1.6. Generating the Partner Relationship Text File	278
Task 5.1.7. Printing a Partner Relationship Report.....	278
Task 5.2. Creating Individual Partners.....	279
Task 5.2.1. Specifying the Partner Name.....	279
Task 5.2.2. Specifying a Location	280

Task 5.2.3. Specifying Partner IDs.....	281
Task 5.2.4. Specifying the Trade Agreement Profile.....	282
Task 5.2.5. Specifying Alternative Information for the Outputs.....	284
Task 5.2.6. Specifying Acknowledgment Profiles.....	287
Task 5.2.7. Generating the Partner Text File	289
Task 5.2.8. Printing a Partner Report.....	289
Creating Definitions Using the Partner Wizard.....	290
Specify How to Control Processing	291
Control Processing Based on New Partner Relationship	292
Control Processing Based on New Individual Partner Definitions.....	294

Configuring Standard ID and Defaults 297

Overview	297
Using Partner Explorer or the Partner Wizard	297
Using the Partner Wizard.....	299
Task 6. Specifying the Source Location	300
Task 6.1. Creating the Source Location.....	303
Task 6.2. Specifying How to Match the Wrapper	304
Task 7. Specifying Processing Requirements	307
Task 7.1. Specifying Processing Options for Partner Definitions.....	308
Task 7.2. Specifying Default Routing Options.....	309
Task 7.3. Specifying Default Trade Agreements	310
Task 7.4. Specifying Default Acknowledgments	311
Task 7.5. Generating the Standard ID Text File	311
Task 7.6. Printing a Standard Identification Report.....	312

Creating Standard ID Using the Partner Wizard	313
Specify How to Control Processing	314
Control Processing Based on New Incoming Wrapper Only.....	315
Special Cases. Standard ID Configurations.....	316
Using the Partner Wizard to Create a Standard ID	316

Validation and User Exits 323

Overview of Validate Routines and User Exits.....	323
Validating Data During Parsing	324
Guidelines for Using Validate Methods during Parsing	325
How to Use Validate Methods during Parsing.....	325
Validating Data during Mapping	327
Validating Data during Generation	328
User Exit Overview	329
User Exit Examples	330
Security User Exit (AUTACK) Overview	331
Processing AUTACK Messages	331
Receiving Secured Messages and Sending Secured Acknowledgments.....	332
Sending Secured Messages and Receiving Secured Acknowledgments.....	334
Checklist of Prerequisite Configurations	335
Checklist of User Exit Functions.....	337
Configuring Definitions to Receive Secured Messages.....	337
Configuration Tasks to Receive Secured Messages and Return Secured Acknowledgment .	338
Writing the User Exits	339
Specifying the Security Document and Security User Exit for Incoming Document(s)	339

Specifying the User Exit to Validate the Digital Signature	344
Specifying the User Exit to Calculate the Hash Value for the Summary Document	346
Specifying the User Exit to Generate the Digital Signature	347
Files for AUTACK Example.....	350
Configuring Definitions to Send Secured Messages	352
Configuration Tasks to Send Secured Messages and Receive Secured Acknowledgment	353
Writing the User Exits	354
Specifying the Security User Exit and the Summary Document for the Outgoing Document	354
Specifying the User Exit to Generate the Digital Signature	357
Specifying the Security Document and User Exit for Incoming Acknowledgment.....	358
Specifying the User Exit to Validate the Digital Signature	363
Example Input Files	364

Extensible Markup Language (XML) Support 367

Overview	367
Validation of XML Data	367
Example XML Translations.....	368
Creating MW Translator Definitions for XML	368
Defining a Document from an XML DTD or XSD	369
Defining an XML Document Manually	373
Specifying Namespaces.....	376
Defining the XML Wrapper	379
Specifying the User Exit to Parse Input.....	381
Completing XML Configurations and Maps and Testing.....	382
Testing XML Input	382

Testing XML Output	384
How the TRM generates XML	384
Loop generation.....	385
Segment generation	385
Element Generation (category of Fixed).....	385
Element generation (category XML Attribut)	385
Element generation (category XML or blank).....	385
Element generation (category XML CDATA).....	386
Element generation (category XML Element)	386
Limitations.....	386
Testing Configurations	387
Information Required for Testing	387
Important Settings for Tests	388
Setting Report Options	388
Setting TRM Options.....	389
Generating Text Files	389
Running a Test.....	390
Reviewing Your Processing Report	394
Header Information	394
Identified Standard Information	395
Processing Wrapper.....	396
Trade Agreement Found.....	397
Translation Output (Documents).....	398
Status of Incoming Wrapper Level 3, Document.....	399

Additional Output	400
Status of Incoming Wrapper Level 2, Functional Group.....	400
Translation Output (Acknowledgments)	401
Status of Incoming Wrapper Level 1, Interchange	402
End of Report	403
Exporting Definitions.....	403

Administration 405

Overview of Administration	405
Creating and Using Various Environments.....	406
Creating Environments	406
Using Environments	409
Moving Environments.....	410
Deleting Environments	411
Moving Information to and from Databases	411
Copying Configurations.....	412
Exporting Configurations	412
Importing Configurations.....	416
Maintaining Databases	419
Transferring MWTranslator Configuration Information to MessageWay	420

Troubleshooting 423

Runtime Exceptions	423
Translator Runtime Module (TRM) Exceptions.....	425
1001 Too few segment or loop repeats.....	426
1002 Too many segment repeats	426

1003 Mandatory entity missing	427
1004 Segment contains too many elements	427
1005 Too many loop repeats	427
1006 Too many composite repeats	428
1007 Too many element repeats.....	428
1101 Element contains invalid character	428
1102 Data element is too long.....	429
1103 Data element is too short	429
1104 Data element contains invalid code value.....	429
1105 Invalid date	430
1106 Invalid time	430
1107 Invalid value	431
1108 Required conditional element missing	431
1109 Invalid delimiter found	431
1110 Unrecognized element tag.....	432
1111 Component separator found in simple element	432
1201 Composite conditional failed	433
1202 Composite conditional failed	433
1301 Segment conditional failed	434
1302 Unrecognized segment	434
1303 Segment out of order.....	434
1304 Data Segment is too long.....	435
1305 Segment conditional failed	435
1401 Invalid Bounded Loop Trailer Segment.....	436

1402 Missing Bounded Loop Trailer Segment	436
1901 A warning was reported during XML parsing of the input	436
1902 An error was reported during XML parsing of the input	437
1903 A fatal error was reported during XML parsing of the input	437
1904 Skipping undefined XML element.....	438
1905 Skipping undefined XML attribute.....	438
2001 Control reference mismatch.....	439
2002 Trailer count does not match actual count.....	439
2003 Undefined sending partner.....	440
2004 Undefined receiving partner.....	440
2005 Routing address is not defined for sending partner	440
2006 Routing address is not defined for receiving partner	441
2009 Invalid wrapper definition	441
2010 Unable to find trade agreement.....	441
2011 Invalid wrapper segment found.....	442
2012 Unable to find the destination location.....	443
2013 Unable to identify the document	443
2015 Defined or received delimiters are invalid.....	443
2016 Wrapper trailer is missing.....	444
4001 Invalid control reference validation method.....	444
4002 Invalid control reference generation method	445
4003 Duplicate control reference.....	445
4004 Control reference is out of sequence	445
4005 Control reference overflow	446

4006 Inconsistent control reference generation.....	446
4007 Control reference number is in use	447
4008 Received control reference is blank	447
5001 Invalid user exit name.....	447
6001 Too few segment or loop repeats.....	448
6002 Too many segment repeats	448
6003 Mandatory entity missing.....	448
6005 Too many loop repeats.....	449
6009 Too many omitted elements.....	449
6010 Reject during generate; processing aborted	449
6011 Error during acknowledgment generation; processing aborted	450
6102 Data element is too long.....	451
6103 Data element is too short	451
6104 Data element contains invalid code value.....	452
6201 Composite conditional failed	452
6202 Composite conditional failed	453
6301 Segment conditional failed	453
6305 Segment conditional failed	454
6401 Invalid Bounded Loop Trailer Segment.....	454
6501 An error occurred during generation; output rolled back.....	454
6502 An error occurred during generation; output aborted.....	455
6503 The Interchange is rejected	455
6504 The Functional Group is rejected.....	455
6505 An error occurred during rollback; the Interchange is rejected	456

7001 Unable to Find Segments in the Document.....	456
9001 Processing aborted by user.....	457
9003 Unable to identify standard of message.....	457
9004 Unable to open file	458
9005 File read error	458
9006 Filename not configured in INI file	458
9007 User defined exception with invalid exception	459
9008 Forced Interchange Reject.....	459
9009 Configuration file is empty.....	459
9010 Unable to load set definition file	460
9011 Invalid condition while loading set	460
9019 TRM failure	460
9020 Unable to load map definition file	461
9021 Invalid element map type	461
9022 Invalid composite map type.....	461
9023 Inbound wrapper/document is not the same as input wrapper/document defined in map	462
9024 Inbound Wrapper Is Not the Same as Input Wrapper Defined in Map.....	462
9030 Unable to load wrapper definition file	463
9040 Invalid Instruction	463
9050 File version mismatch	463
9060 Missing Profile.....	464
9070 Unable to open control reference database	464
9080 Unable to open control reference table	464

9081 Unable to read control reference database	464
9082 Unable to write to the control reference database.....	465
9083 Unable to update control reference database	465
9090 Invalid document/wrapper/map definition file.....	465
9100 System Memory Error	465
9101 Fatal System Exception	466
9110 Security Reject	466
9120 Unable to load user exit DLL.....	467
9130 Unable to register user exit DLL.....	467
9140 Security Reject	467
9150 Security Abort	468
10200 Internal System Error	468
15001 Security Reject	468
20001 or higher User defined exception	469
Understanding the Processing Report	469
Status Information	470
Data Element Store Information.....	471
File Name Information.....	471
Processing Trace Information	472
Parse Errors	473
Parsing Process	474
Viewing the Parsed Input Data	474
Viewing the Contents of the Data Element Store.....	475

Generate Errors.....	483
Generating Configurations.....	483
Generating Data.....	485

General Reference 487

Title Bar	487
Toolbar	487
Menus and Task Icons.....	487
File Menu (Workbench).....	488
Edit Menu (Workbench)	489
View Menu (Workbench).....	489
Generate Menu (Workbench).....	490
Tools Menu (Workbench).....	490
Check Errors Menu (Workbench)	491
Windows Menu (Workbench).....	491
Help Menu (Workbench)	491
Copy Command.....	492
Enter the Destination for Copy Dialog Box	493
Select Environment or Database	493
Environment Value	494
Database Value	494
Rename Definition	495
Copy Related Definitions	495
Procedures (Copy Command).....	495

Export Command	497
Procedures (Export Command)	498
Generate Command.....	499
Procedures (Generate Command).....	500
Import Command.....	501
Import Definitions Window	503
Transfer FileName (Import Definitions).....	503
Import All	504
Definitions in Transfer File.....	504
Prompt for Rename	504
View Segments, Composites and Elements.....	504
Import Button	505
Procedures (Import Definitions).....	505
Load Standards Command	506
Procedures (Load Standards Command)	507
Options Command	507
(Modify Options) General Page.....	508
Report Options.....	508
Only Print Reports If Errors (Report Options).....	509
Always Print Reports (Report Options).....	509
Print DES (Report Options)	509
Print File Names (Report Options).....	509
Print Processing Trace (Report Options).....	509
HyperText Color	512

Reported Exceptions Limit	512
Procedures (Modify Options, General Tab)	512
(Modify Options) Directories Page	514
Root Directory	515
Details Button	515
Database (Subdirectories)	516
Configuration (Subdirectories)	516
Input (Subdirectories)	517
Output (Subdirectories)	518
Report (Subdirectories)	518
Files (Subdirectories)	519
Procedures (Modify Options, Directories Tab)	519
(Modify Options) User Exits Page	520
User Exit DLLs	520
Procedures	521
Pack Database Command	521
Pack Database	521
Procedures (Pack Database Command)	522
Print Command	522
Print Preview Command	524
Print to PDF Command	525
Select Environment Command	525
Select Environment	526
Add Button	526

Environment Name	527
Select DB Directory	527
Remove Button	527
Procedures (Select Environment Command)	527
Data Explorer Reference	531
Data Explorer Window	531
To Open a Properties Window.....	532
To Open the Data Explorer Window.....	532
Acknowledgment Profile Window	532
Acknowledgment Profile Name	533
(Acknowledgment Profile) Input.....	533
Inbound Wrapper	534
(Acknowledgment Profile) Maps	534
Acknowledgment Map.....	534
Wrapper Map.....	535
(Acknowledgment Profile) Options.....	535
Create Acknowledgment	535
Report Doc Status	536
(Acknowledgment Profile) Summary Document	537
Summary Document Map.....	537
Security User Exit	538
Composite Window.....	538
Composite Name	539
Description.....	539

Seq (Components)	539
Ele ID (Components)	539
Description (Components).....	540
Blank Column	540
Field (Components).....	540
Rqmt (Components).....	541
Type (Components)	542
Min (Components).....	542
Max (Components)	542
Procedures (Composite)	542
Composite Properties Window	549
Procedures (Composite Properties).....	549
(Composite Properties) Conditions.....	551
Seq	551
Sub Seq	552
Condition.....	552
Element Sequence Numbers	552
(Composite Properties) General	553
Category.....	553
Purpose	554
(Composite Properties) Where Used	554
Where Used (Composite Properties).....	555
Cross Reference Window	556
Description.....	557

Input.....	557
Output.....	557
Procedures (Cross Reference).....	557
Document Window	559
Report Segment Detail (Document Report Options Dialog Box).....	562
Show Segment Conditionals (Document Report Options Dialog Box)	562
Show Validate Methods (Document Report Options Dialog Box)	562
Document Name	562
Description (Document).....	562
Functional Group	562
L/S	563
Level	563
Area.....	563
Seq	564
Tag.....	564
Description (Segment)	564
Blank Column	564
Rqmt.....	568
Max Occ	568
Procedures (Document).....	568
Document Properties Window	575
Procedures (Document Properties)	575
(Document Properties) General.....	577
Category.....	578

Document Type.....	578
Validate Element Codes.....	579
Ack Level	579
Purpose	579
Ack XRef	579
Edibasic Edit Window.....	580
Variables (Edibasic Edit).....	582
Start (Edibasic Edit).....	583
GetNext (Edibasic Edit)	584
Stop (Edibasic Edit)	584
Condition (Edibasic Edit).....	584
MapEle (Edibasic Edit)	585
Validate (Edibasic Edit)	585
ValidateAll (Edibasic Edit)	585
Procedures (Edibasic Edit)	585
Element Window.....	587
Element Name	588
Description (Element).....	588
Min	588
Max	589
Type	589
Code	589
Description (Element Code).....	589
Partition	590

Procedures (Element)	590
Element Properties Window	595
(Element Properties) General	596
Category	596
Purpose	598
(Element Properties) Where Used	599
Where Used	599
Map Window	600
Map Name	601
Description	601
Map Type	601
Source Document	602
Source Wrapper	603
Destination Document	604
Selected Item Definition Detail	604
Mapping Source	605
DOC	605
WRAP	605
SSEDOC	606
SSEWRAP	606
Mapping Destination (Map Window)	606
Source Of Mapped Item (Map Window)	606
Procedures (Map)	607

Segment Window	616
Segment Name	623
Description	623
Seq	623
Ele ID	623
Description (Element)	624
Blank Column	624
Tag	624
Field	625
Rqmt	626
Rep	626
Type	626
Min	626
Max	626
Procedures (Segment)	626
Segment Properties Window	633
Procedures (Segment Properties)	633
(Segment Properties) General	635
Category	636
Purpose	637
(Segment Properties) Conditions	637
Seq	637
Sub Seq	638
Condition	638

Element Sequence Numbers	638
(Segment Properties) Where Used	639
Where Used	639
(Segment Properties) XML	640
Namespace	640
Standard Version Window	641
Standard Name	642
Enter Version Name	642
Description	642
Type (Types)	642
Base Type (Types)	643
Character Set (Types)	644
Format (Types)	645
Procedures (Standard Version)	645
Standard Version Properties Window	649
Category	650
Validate Element Codes	650
Characters to Ignore Between Segments	651
Procedures	651
Trade Agreement Profile Window	652
(Trade Agreement Profile) Matching Fields	653
Trade Agreement Profile Name	653
Identified Standard	654
Doc Id (Input Field Values)	654

Version (Input Field Values).....	654
Release (Input Field Values).....	654
Agency (Input Field Values)	654
Assoc (Input Field Values).....	655
(Trade Agreement Profile) Input.....	655
Inbound Document	655
Security Document	656
Security User Exit	656
(Trade Agreement Profile) Options.....	656
Action.....	657
Error Action.....	658
(Trade Agreement Profile) Outputs	658
Outputs List box	659
Add Button.....	659
Remove Button	659
Properties Button	659
Move Up Button	659
Move Down Button.....	659
Trade Agreement Output Properties Window	660
(Trade Agreement Output Properties) General	660
Sequence.....	661
Output #.....	661
Document Map (Trade Agreement Output Properties)	661
Wrapper Map (Trade Agreement Output Properties).....	661

(Trade Agreement Output Properties) Summary Document	662
Summary Doc Map	662
Security User Exit	663
(Trade Agreement Output Properties) Alias	663
Alias Type	664
Wrapper Window.....	664
Wrapper Name	666
Description (Wrapper)	666
Blank Column	666
Lvl	666
Seq	666
Seg Tag	667
Description (Segment)	667
Rqmt.....	667
Header (Contents).....	667
Trailer (Contents).....	667
Procedures (Wrapper)	667
Wrapper Properties Window	674
Procedures (Wrapper Properties).....	674
(Wrapper Properties) General	679
Category.....	680
Validate Element Codes.....	680
Binary (IO Mode).....	681
Text (IO Mode).....	681

Pre-Process Method	682
Post-Process Method.....	682
Ack XRef	682
(Wrapper Properties) Service Characters	683
Segment Terminator.....	684
Tag Delimiter.....	684
Element Delimiter	684
Component Delimiter	685
Repetition Separator.....	685
Release Character	686
Decimal Mark.....	686
(Wrapper Properties) StdID Match	687
Default Standard ID Match Criteria	688
Operator (Match Criteria).....	689
Offset (Match Criteria).....	689
Segment (Match Criteria)	689
Field (Match Criteria).....	689
SubField (Match Criteria).....	690
Value (Match Criteria).....	690
Up Button.....	690
Down Button.....	690
Modify Button.....	690
New Button	690
Delete Button.....	691

Edibasic Reference	693
Introduction to Edibasic	693
Edibasic Syntax Requirements	693
Keywords	693
Variable Data Types	695
Variable Names	696
Variable Types and Declaration	696
Variable Lifetime and Scope	697
Literals.....	697
Element Data Types	698
Data Element Names	698
Operators.....	704
Expressions and Operators	704
Operator Syntax	706
Not operator	706
+ operator	706
- operator	707
* operator	707
/ operator.....	707
\ operator.....	707
Mod operator	708
& operator	708
= operator	708
<> operator	708

< operator	709
> operator	709
<= operator	709
>= operator	709
And operator.....	710
Or operator	710
Xor operator.....	710
Eqv operator	711
Imp operator	711
Method Syntax.....	712
Method Condition as Integer.....	712
Method GetNext as Integer	713
Method MapEle as String	714
Method Start.....	715
Method Stop	715
Method Validate as Integer	716
Method ValidateAll as Integer	717
Statement Syntax.....	717
Const statement	718
Dim statement	718
Do-Loop statement.....	719
For Each statement.....	720
Global statement	721
If statement	723

Let statement.....	724
Mid\$ statement	724
Print statement.....	725
Rem statement	726
Select Case statement	726
While statement	728
Function Syntax.....	728
Abs function.....	728
Asc function	729
Avg function.....	729
Exit statement.....	729
For statement	730
Chr\$ function	731
Component\$ function.....	731
Convert\$ function	732
Count function	732
DateSerial function	733
Day function.....	733
Element\$ function	734
Exception function	734
Field\$ function	735
Format\$ function	737
Hour function.....	740
InitOcc function.....	741

InStr function	741
IsNull function.....	742
IsNumeric function	742
LCase\$ function.....	743
Left\$ function.....	743
Len function	744
LTrim\$ function.....	744
Max function.....	744
Mid\$ function	745
Min function	745
Minute function	746
Month function.....	746
NextOcc function	747
Now function	748
Occurrence function	748
ResetOcc function.....	749
Right\$ function	750
RTrim\$ function	751
Second function	751
SetField function.....	751
Space\$ function	753
Str\$ function	754
StrComp function.....	754
String\$ function	754

Sum function.....	755
TimeSerial function.....	755
Trim\$ function	756
UCase\$ function.....	756
User function	756
Val function.....	757
Value function.....	757
VarType function	758
Weekday function.....	759
Xref\$, XrefR\$ functions.....	759
Year function.....	760
Test Reference	761

Test Toolbar and Icons	761
Toolbar.....	761
Output Icons	761
Procedures.....	762
Test Menu and Task Icons.....	762
Force Text (Test Menu).....	763
Auto-delete Outputs (Test Menu)	763
Procedures.....	763
Test Window	764
Input List (Test).....	766
Input Filename (Input List).....	767
Location (Input List)	768

Bytes Read (Input List)	768
Status (Input List)	769
Output List (Test)	769
Output Filename (Output List)	769
Location (Output List)	770
Bytes Written (Output List)	770
Status (Output List)	770
Export Translation Definitions	770
Procedures	771

Partner Explorer Reference 775

Partner Explorer Toolbar	775
Partner Explorer Window	776
Finding Things With Partner Explorer	776
Partner Explorer folders	777
Groups	777
To Open the Partner Explorer Window	777
Partner Window	778
Procedures	778
(Partner) General Page	786
Add Partner Dialog Box	786
Partner Name	787
Location	787
(Partner) IDs Page	788
ID (IDs Page)	789

Qual (IDs Page).....	789
Int ID (IDs Page)	789
Int ID2 (IDs Page)	790
More Button (IDs Page, Partner)	790
(Partner) IDs (cont.) Page.....	791
ID (IDs (cont.) Page)	791
Qual (IDs (cont.) Page)	792
Int ID (IDs (cont.) Page).....	792
Int ID2 (IDs (cont.) Page).....	793
Remove Button	793
(Partner) Trade Agreements Page	794
Trade Agreements List.....	795
Name (Trade Agreements List)	795
Group (Trade Agreements List)	795
Standard (Trade Agreements List)	795
Version (Trade Agreements List)	796
Document (Trade Agreements List).....	796
Release (Trade Agreements List)	796
Agency (Trade Agreements List).....	796
Assoc (Trade Agreements List)	797
Continue Search for TA on Standard ID	797
Add Button.....	797
Remove Button	797
Modify Group Button.....	797

Options Button.....	798
Adjust Output Sequences Dialog Box	798
Edit Profile Button (Partner Window, Trade Agreements Page)	798
(Partner) Acknowledgments Page	799
Acknowledgments List	799
Name (Acknowledgments List)	799
Standard (Acknowledgments Listbox)	800
Level (Acknowledgments List)	800
Add Button (Partner Window, Acknowledgments Page)	800
Select Acknowledgment Profile Dialog Box (Partner Window, Acknowledgments Page)	801
Remove Button (Partner Window, Acknowledgments Page).....	801
Options Button (Partner Window, Acknowledgments Page)	801
Edit Profile Button (Partner Window, Acknowledgments Page)	802
(Partner) Groups Page.....	802
Groups List box	804
Add Button (Partner Window, Groups Page)	804
Remove Button (Partner Window, Groups Page).....	804
Edit Button (Partner Window, Groups Page).....	804
(Partner) Alias Page.....	804
Alias List	806
Alias Type (Alias List).....	806
Alias Partner (Alias List)	806
Location (Alias List)	807
Add Button.....	807

Remove Button	807
Edit Button	807
Partner Trade Agreement Options Window	807
Output.....	807
Procedures (Partner Trade Agreement Window Options)	808
(Partner Trade Agreement Options) Sending Partner Page.....	810
Partner Name (Sending Partner)	810
Location (Sending Partner)	811
Override Location	811
ID (Sending Partner).....	811
Qual (Sending Partner)	811
Int ID (Sending Partner)	811
Int ID2 (Sending Partner)	811
(Partner Trade Agreement Options) Recipient Partner Page	812
Partner Name (Recipient Partner)	812
Location (Recipient Partner)	813
Override Location	813
ID (Recipient Partner)	813
Qual (Recipient Partner)	813
Int ID (Recipient Partner)	813
Int ID2 (Recipient Partner)	813
(Partner Trade Agreement Options) Misc Page.....	814
Request Acknowledgment.....	815
Test	815

Service Characters	815
Segment Terminator	815
Tag Delimiter	816
Element Delimiter	816
Repetition Separator.....	816
Component Delimiter	816
Release Character	817
Decimal Mark.....	817
Document Level Break.....	817
Acknowledgment Expected	817
Output Validation	817
Partner Acknowledgment Options Window	818
Procedures	818
(Partner Acknowledgment Options) General Page.....	820
Send Outputs	821
Copy from input stream.....	821
Segment Terminator.....	821
Tag Delimiter.....	821
Element Delimiter	822
Repetition Separator.....	822
Component Delimiter	822
Release Character	822
Decimal Mark.....	823

Partner Relationship Window	823
Procedures	823
(Partner Relationship) General Page.....	826
Add Partner Relationship Dialog Box.....	827
Partner Relationship Name.....	827
(Partner Relationship) Sending Partner Page	828
Sender Name	828
Location (Sending Partner Page)	828
ID (Sending Partner Page).....	829
Qual (Sending Partner Page).....	829
Int ID (Sending Partner Page).....	829
Int ID2 (Sending Partner Page)	829
More Button (Sending Partner Page, Partner Relationship)	829
(Partner Relationship) Sending Partner (cont.) Page.....	830
ID (Sending Partner (cont.) Page)	830
Qual (Sending Partner (cont.) Page)	830
Int ID (Sending Partner (cont.) Page).....	831
Int ID2 (Sending Partner (cont.) Page).....	831
Remove Button	831
(Partner Relationship) Recipient Partner Page	832
Recipient Name.....	832
Location (Recipient Partner Page)	832
ID (Recipient Partner Page).....	833
Qual (Recipient Partner Page)	833

Int ID (Recipient Partner Page)	833
Int ID2 (Recipient Partner Page)	833
More Button (Recipient Partner Page, Partner Relationship)	833
(Partner Relationship) Recipient Partner (cont.) Page.....	834
ID (Recipient Partner Page).....	834
Qual (Recipient Partner Page)	834
Int ID (Recipient Partner Page)	835
Int ID2 (Recipient Partner Page)	835
Remove Button	835
(Partner Relationship) Trade Agreements Page	836
Name (Trade Agreements List)	836
Standard (Trade Agreements List)	836
Version (Trade Agreements List)	837
Document (Trade Agreements List).....	837
Release (Trade Agreements List)	837
Agency (Trade Agreements List)	837
Assoc (Trade Agreements List)	837
Continue Search for TA on Recipient Partner	838
Add Button.....	838
Remove Button	838
Options Button.....	838
Edit Profile Button.....	838
(Partner Relationship) Acknowledgments Page.....	839
Acknowledgments List	840

Name (Acknowledgments List)	840
Standard (Acknowledgments List)	840
Level (Acknowledgments List)	841
Add Button (Acknowledgments).....	841
Remove Button (Acknowledgments).....	841
Options Button (Acknowledgments)	841
Edit Profile Button (Acknowledgments)	841
Partner Relationship Trade Agreement Options Window	842
Output.....	842
Procedures	842
(Partner Relationship Trade Agreement Options) Sending Partner Page.....	845
Partner Name (Sending Partner)	845
Location (Sending Partner)	846
Override Location	846
ID (Sending Partner).....	846
Qual (Sending Partner)	846
Int ID (Sending Partner)	846
Int ID2 (Sending Partner)	846
(Partner Relationship Trade Agreement Options) Recipient Partner Page	847
Partner Name (Recipient Partner)	847
Location (Recipient Partner)	848
Override Location	848
ID (Recipient Partner)	848
Qual (Recipient Partner)	848

Int ID (Recipient Partner)	848
Int ID2 (Recipient Partner)	848
(Partner Relationship Trade Agreement Options) Misc Page	849
Request Acknowledgment	850
Test	850
Service Characters	850
Segment Terminator	850
Tag Delimiter	851
Element Delimiter	851
Repetition Separator	851
Component Delimiter	851
Release Character	852
Decimal Mark	852
Document Level Break	852
Acknowledgment Expected	852
Output Validation	852
Partner Relationship Acknowledgment Options Window	853
Procedures	853
(Partner Relationship Acknowledgment Options) General Page	854
Send Outputs	855
Copy from input stream	855
Segment Terminator	855
Tag Delimiter	855
Element Delimiter	856

Repetition Separator.....	856
Component Delimiter	856
Release Character	856
Decimal Mark.....	857
Standard ID Window	857
Procedures.....	857
(Standard ID) General Page.....	860
Location Name	861
Standard Name	861
Match Criteria	863
Operator (Match Criteria).....	864
Offset (Match Criteria).....	864
Segment (Match Criteria)	864
Field (Match Criteria).....	864
SubField (Match Criteria).....	865
Value (Match Criteria).....	865
Up Button.....	865
Down Button.....	865
Modify Button.....	865
New Button.....	865
Delete Button.....	866
(Standard ID) Partners Page.....	866
Sending Partner (Partner Definition Required)	867
Recipient Partner (Partner Definition Required)	867

Name (Default Sending Partner).....	867
Name (Default Recipient Partner)	867
(Standard ID) Routing Page	868
Default Output Routing	868
Data Field (Default Output Routing).....	869
Partner ID (Data Field, Default Output Routing)	869
Partner Int ID (Data Field, Default Output Routing)	869
Source Address (Default Output Routing)	869
Use Source before Data Field (Default Output Routing)	869
Use Source before Partner Location (Default Output Routing).....	870
Default (Default Output Routing)	870
Source (Default Output Routing)	870
Destination (Default Output Routing)	870
Default Acknowledgment Routing.....	870
Data Field (Default Acknowledgment Routing)	871
Partner ID (Data Field, Default Acknowledgment Routing).....	871
Partner Int ID (Data Field, Default Acknowledgment Routing)	871
Source Address (Default Acknowledgment Routing)	871
Use Source before Data Field (Default Acknowledgment Routing)	871
Use Source before Partner Location (Default Acknowledgment Routing).....	872
Default (Default Acknowledgment Routing)	872
Source (Default Acknowledgment Routing)	872
Destination (Default Acknowledgment Routing).....	872

(Standard ID) Trade Agreements Page.....	873
Trade Agreements List	874
Name (Trade Agreements List)	874
Closed (Trade Agreements List)	874
Standard (Trade Agreements List)	874
Version (Trade Agreements List)	875
Document (Trade Agreements List).....	875
Release (Trade Agreements List)	875
Agency (Trade Agreements List).....	875
Assoc (Trade Agreements List)	875
Add Button.....	876
Select Trade Agreement Profile.....	876
Remove Button	876
Options Button.....	877
Edit Profile Button.....	877
(Standard ID) Acknowledgments Page.....	878
Acknowledgments List	878
Name (Acknowledgments List)	879
Standard (Acknowledgments List)	879
Level (Acknowledgments List)	879
Add Button (Standard ID Window, Acknowledgments Page)	879
Select Acknowledgment Profile Dialog Box (Standard ID Window, Acknowledgments Page).....	880
Remove Button (Standard ID Window, Acknowledgments Page).....	880
Options Button (Standard ID Window, Acknowledgments Page)	880

Edit Profile Button (Standard ID Window, Acknowledgments Page)	881
Standard ID Trade Agreement Options Window	881
Output.....	881
Procedures.....	881
(Standard ID Trade Agreement Options) Sending Partner Page.....	884
Partner Name (Sending Partner)	884
Location (Sending Partner)	885
Override Location	885
ID (Sending Partner).....	885
Qual (Sending Partner)	885
Int ID (Sending Partner)	885
Int ID2 (Sending Partner)	885
(Standard ID Trade Agreement Options) Recipient Partner Page	886
Partner Name (Recipient Partner)	886
Location (Recipient Partner).....	886
Override Location	887
ID (Recipient Partner)	887
Qual (Recipient Partner)	887
Int ID (Recipient Partner).....	887
Int ID2 (Recipient Partner).....	887
(Standard ID Trade Agreement Options) Misc Page	888
Request Acknowledgment.....	889
Test	889
Service Characters	889

Segment Terminator	889
Tag Delimiter.....	890
Element Delimiter	890
Repetition Separator.....	890
Component Delimiter	890
Release Character	891
Decimal Mark.....	891
Document Level Break	891
Acknowledgment Expected	891
Output Validation	891
Standard ID Acknowledgment Options Window	892
Procedures	892
(Standard ID Acknowledgment Options) General Page.....	893
Send Outputs (Partner Acknowledgment Options).....	893
Copy from input stream.....	894
Service Characters (Partner Acknowledgment Options)	894
Segment Terminator	894
Tag Delimiter.....	894
Element Delimiter	895
Repetition Separator.....	895
Component Delimiter	895
Release Character	895
Decimal Mark.....	896

Partner Wizard	897
Overview	897
Partner Wizard Window.....	898
To Open the Partner Wizard Window	898
(Partner Wizard) Specify How to Control Processing Page.....	899
Control Processing Based on NEW Partner Relationship	900
Generate Acknowledgment (Partner Relationship)	900
Control Processing Based on NEW Individual Partner.....	900
Generate Acknowledgment(s) (Partner)	901
Generate Document(s) (Partner).....	901
Control Processing Based on NEW Incoming Wrapper	901
Generate Acknowledgment(s) (Standard ID/Wrapper).....	901
(Partner Wizard) Specify How to Identify the Incoming Standard Page.....	902
Inbound (Source) Location.....	902
Inbound Wrapper Standard	903
Back Button.....	903
Next Button.....	903
Cancel Button.....	903
(Partner Wizard) Match Criteria Page.....	904
Match Criteria	905
Up Button.....	905
Down Button.....	905
Modify Button.....	905
New Button.....	905

Delete Button.....	905
(Partner Wizard) Partner Relationship Page.....	906
Relationship Name.....	907
Sender Name	907
Location	907
Interchange ID and Qualifier.....	907
Functional Group ID.....	907
(Partner Wizard) Partner Relationship (cont.) Page	908
Recipient Name.....	909
Location	909
Interchange ID and Qualifier.....	909
Functional Group ID.....	909
(Partner Wizard) Select Trade Agreement Page	910
Trade Agreement List.....	910
Continue Search for TA on.....	910
(Partner Wizard) Partner Page.....	911
Partner Name.....	912
Location	912
Interchange IDs and Qualifier	912
Functional Group IDs and Qualifier	912
(Partner Wizard) Select Trade Agreement Profile Page.....	913
(Partner Wizard) Trade Agreement Options Page.....	914
(Partner Wizard) Select Acknowledgments Page.....	915
Acknowledgments List	915

Add Button.....916

Remove Button916

Options Button.....916

(Partner Wizard) Acknowledgment Options Page917

(Partner Wizard) Confirm Update Page918

 Partner Relationship918

 Partner919

 Standard ID920

 Finish Button.....921

 Back Button.....921

 Cancel Button.....921

Appendix Licenses 923

Xerces.....923

 Apache License924

 Apache Xerces Notice929

Glossary of Terms 931

Index 935

This page intentionally blank

Basic Information

Typographical Conventions

Before you start using this guide, it is important to understand the terms and typographical conventions used in the documentation.

The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information
Special Bold	<ul style="list-style-type: none"> ▪ Items you must select, such as menu options, command buttons, or items in a list ▪ Type of note ▪ Values you must type, constants
<i>Emphasis</i>	<ul style="list-style-type: none"> ▪ Titles of books ▪ Important word ▪ Captions for figures ▪ Variable expressions such as parameters
Monospace	Code samples
CAPITALS	Names of keys on the keyboard. for example, SHIFT, CTRL or ALT.
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another, for example, CTRL+P or ALT+F4.

The following formatting is used to explain command syntax:

Formatting convention	Type of Information
Special Bold	Values you must type, constants
<i>Emphasis</i>	Variable expressions such as parameters
Monospace	Code samples
[]	Brackets enclose optional parameters
{ }	Braces or curly brackets enclose required parameters
	Bar separates options within brackets or braces
(...)	Indicates options may repeat

Purpose of the MW Translator Workbench

This application is an Electronic Data Interchange (EDI) processor developed for the environment of distributed applications. It provides EDI services for validation and routing of either the wrapper or the wrapper and contents, as well as for full translation. The program executes in two modes: test and production. In test mode, it runs on a PC under Windows as the Workbench. The Workbench allows you to configure definitions, identify processing requirements, and run tests. In production mode, it can run on various target platforms and is typically integrated with other software to provide full-featured EDI production processing or EDI services for specific applications. We refer to the production mode environment as the target environment. The Translator Runtime Module (TRM) is that part of the target environment that does EDI processing. MessageWay that runs on Windows and MessageWay that runs on UNIX/Linux are examples of different TRMs.

Audience for MW Translator

In order to use this material effectively, you must understand EDI in general, which means issues about mapping and partnerships. To use Edibasic effectively, you should understand basic programming techniques. In addition, you must already be familiar with Windows.

What You Can Find Here

The information presented here is to help you understand and use this program effectively. It includes a combination of user's guide and reference material. This information does not contain a complete example, although there are basic configuration tasks that you can follow. The user information is a combination of explanations of processing flow and instructions to help you perform specific tasks. For example, if you are translating, you must first map your documents. The reference materials explain the various configuration windows for the Workbench Data Explorer and Partner Explorer and the components of Edibasic. The reference material for Edibasic includes a description of valid operators, methods, statements, and functions.

NOTE: If you are looking for examples that describe the process from the beginning of configuration to the end of testing, you will find them in the manual entitled *MW Translator Workbench Tutorial*.

Basic Features

The Workbench includes many features that improve usability and productivity.

- Easy partnership configuration
 - Partner Wizard to help you configure processing control by adding partner definitions and standard ID definitions easily and quickly
 - Flexible partner configurations that support partner relationships as well as the full capabilities of previous versions of the software
 - Extensive routing determination strategies
- Support for more electronic commerce features (see EXAMPLES directory):
 - Summary documents support EDIFACT AUTACK and X12 980
 - Security User Exits (refer to the AUTACK example)
 - XML syntax support for inbound and outbound documents (refer to the XML examples)
 - Validate input and optionally output against configurations
- Integrated utilities and tools for:
 - Moving configurations (import, export, copy)
 - Loading public standards definitions as you need them directly from disk (load standards)
 - Maintaining definitions database
 - Debugging enhancements include various types of trace information, which prints more information on the translation report about exactly what MW Translator does during processing.
- Browser format and functionality allowing users to see relationships at a glance with collapsible views
 - Multi-window concurrent access enables quick referencing
 - Data Explorer provides access to standards definitions, maps, trade agreements, and acknowledgments
 - Partner Explorer provides access to standards ID (locations), partner, partner relationships, and group definitions
- Testing capabilities
 - 32-bit architecture that allows for testing of larger segments (>64K) and very large documents
 - Test TRM window allows you to queue and run multiple files and review information about your tests
 - Debugging feature to print traces on processing reports
- Reporting
 - Processing/translation reports indicate specific information about the source location, what configurations were used by the TRM to process the data, and date and time stamps that indicate the beginning and end of processing

- Definitions reports provide high-level and detail views
- Select output to printer, screen or PDF file
- Multiple database environments allows users to easily switch from one database environment to the other using selections from the task bar
- Database environments are integrated with an MW Translator Operator program that provides controlled access to and separate control of the configurations and queries for the target, production environment

Getting Started

Technical Support

The MessageWay Technical Support hub is an information and diagnostic center available for customers to:

- Obtain advice on proper product installation, configuration, and operation
- Report any product problems and receive timely resolutions
- Request a software enhancement
- Request software updates
- Inquire about software release contents and status
- View publications
- See how to contact Technical Support
- See hours of availability for Technical Support

To visit the MessageWay Technical Support hub, please follow the below link:

<https://www.progress.com/support/messageway>

The Technical Support Web site is available 24/7, portions of which require a valid Progress ID. If you have not already done so, you can follow the instructions in the following URL to obtain a valid Progress ID:

<https://knowledgebase.progress.com/articles/Article/how-to-create-a-progress-id>

Related Information

There are several different types of related and supporting information available in varying formats and locations. Please also review the following:

Title	Level	Description
Progress Support Web site	(Web site for additional customer support) beginner to advanced	https://www.progress.com/support/messageway This is the Technical Support web site.
MessageWay Knowledgebase	Beginner to advanced	https://knowledgebase.progress.com This web site contains answers to frequently asked questions.
<i>MW Translator Workbench Tutorial</i>	(for testing) beginner to advanced	Document available as a .PDF file on installation medium; contains complete instructions for users to create EDIFACT and X12 translation examples as they familiarize themselves with the Workbench.
<i>MW Translator Operator Guide and Reference</i>	(for EDI production) beginner to advanced	Document available as a .PDF file; contains instructions on how to use the Operator program to configure partners, implement and use auditing and reconciliation, and transfer files from a test environment to a MessageWay production environment.
<i>MessageWay User's Guide and Reference</i>	(for messaging using MessageWay) beginner to advanced	Document available as a .PDF file on installation medium, contains user and reference information for administrators and operators who will use the MessageWay Manager.
<i>MW Translator User Exits Programming Manual</i>	C or C++ programmer	Document available as a .PDF file on installation medium, contains instructions on writing code for one of the seven types of user exits.
<i>MW Translator API Reference and Programming Manual</i>	advanced programmers	Document available as a .PDF file, contains information to allow programmers to integrate the software into an application to provide EDI type validation, translation and acknowledgments.

What Features are Supported

Although this program converts data file formats, it also handles complex syntax, such as those often found in EDI. It fully supports X12 and EDIFACT syntactic requirements and contains the latest document definitions for the primary releases. It also supports SWIFT syntax, but does not currently contain any document definitions. Users can always enter definitions they may require that are not installed with the software.

The following features are part of the current release:

- X12 standards definitions: 003010 through the latest release (primary releases)
- EDIFACT standards definitions: 91.2, S93.A through the latest release (primary releases)
- SWIFT syntax support
- XML syntax support
- Proprietary standards (entered by user)
- Visual (GUI) mapping
- Edibasic mapping language
- Concurrent parallel translations
- Small and large interchanges
- One-to-many documents mapping
- Compliance checking on input document
- Configurable element code validation
- Repeating elements and composite elements
- Document-level reject
- Backward acknowledgments
- Proprietary acknowledgments
- Summary documents (forward acknowledgments)
- Specific partner trade agreements
- Open trade agreements with unknown sender
- Open trade agreements where neither sender nor recipient are known
- Closed trade agreements
- Partner Wizard
- User validation routines for additional compliance checking
- Global variables available for entire input stream
- User exit interfaces
- Authentication and acknowledgment support
- Multiple database environments

System Requirements and Recommendations

To run the MW Translator Workbench and Operator program, refer to the requirements list in the *MW Translator Installation Guide*.

Utilities

There are utilities available in the **Workbench\utils** subdirectory as follows:

Utility	Description
DTD2TRN	Converts an XML DTD to a MessageWay TRN file, which can then be imported to MessageWay to create appropriate configurations for the XML document.
XSD2TRN	Converts an XML Schema to a MessageWay TRN file, which can then be imported to MessageWay to create appropriate configurations for the XML document.

Working Examples

Several different types of working examples are included with the install. Sometimes the definitions are already loaded and all you have to do is submit an input file to run a test. Sometimes the definitions must be loaded from transfer files contained in the **WORKBENCHEXAMPLES** directory. You can also enter your own X12 and EDIFACT examples using the instructions in the *MW Translator Workbench Tutorial*.

The following working examples are currently included when you install the software (refer to the **README** file with the example for instructions to load the definitions and run the example):

- A translation example in the **X850TEST** subdirectory
- Several user exit routines are located in the following subdirectories (For more information about user exits and the examples, refer to the *MW Translator User Exits Programming Manual*):

User Exit Type	Subdirectory Location
Audit	\AUDIT
Edibasic	\VALELE
Security	\AUTACK
Pre-Processing	\ADDTAG
Post-Processing	\REMSPC

- Visual C++ and C++ Builder projects to build user exits are in the **Projects** subdirectory. All user exit examples are thread safe and may be used by the Workbench or with a MessageWay Windows agent
- Source code for user exits is located in the **SOURCE** subdirectory
- Two SWIFT examples are located in the **SWIFT** subdirectory
- Two XML examples are located in the **XML** subdirectory

Additional Standards

MW Translator installs the X12 and EDIFACT standards that it supports. Regarding new standards becoming available that MW Translator does not currently support, users should contact MessageWay Technical Support.

MW Translator Database Environments

You install the Workbench on your personal computer using the install package. For an explanation of your options, please read the *MW Translator Installation Guide*. You also use the Workbench install to create additional database environments to store your configurations.

Installing Additional Database Environments

When you first install MW Translator, you install a default database environment. You may find that you want to maintain separate database environments with their own configurations and standards definitions for a Workbench. To do so, you select the option **New Database Environment** from the Workbench install. The install will not create another copy of the Workbench, but allows you to install a different environment with its own set of files.

Converting Previous Versions of the MW Translator Configuration Databases

If you have current Workbench databases for versions 4.0.1 or higher, the install will attempt to upgrade the structure. When you open the Workbench, you must answer if you want to convert them. You can run multiple versions of MW Translator simultaneously, so if do not convert your current database, the installation programs creates a new database and supporting environment, which contain only the default configurations.

IMPORTANT: To upgrade Workbench databases that are earlier than version 4.0.1, you must upgrade incrementally to 4.0.1 before you install version 5.0. For example, if you are running versions 2.0 or 3.0, you must first upgrade to 4.0.1, and then to 5.0.

Whenever you import files created with earlier versions of the software, the Workbench will load them to the new database structure.

IMPORTANT: There is no backward compatibility with earlier versions of the Workbench. In addition, the **Import** command does not support files generated from Edikit versions earlier than 2.0, and MessageWay does not support files generated from Edikit versions earlier than 4.0.1. Although the import may work, there is no guarantee on the results.

Planning and Configuration

Overview

The information here is primarily for the Workbench. In reality, you're planning and configuration efforts will affect more than just the Workbench environment. The Workbench is part of a larger EDI application. It allows you to develop and test configurations. You use the tested configurations in a production environment running MessageWay® on a Windows®, Linux or UNIX® system.

You should understand the following issues in order to better plan and configure your Workbench environment:

- Determining your trading requirements: answering basic questions that allow you to configure your system to support your trading requirements
- Using a template approach for new configurations: developing templates based on what you learn by doing your first few configurations to make adding new trading partners easy
- Moving configuration files from development to production: developing a strategy to integrate testing with production

Determining Your Trading Requirements

This approach to translation is bold and provides nearly limitless possibilities. How you define your configurations depends on your trading requirements. Before you can configure anything, you must know what service and trading relationship you require.

Begin by determining your trading requirements. Since requirements will vary for different businesses and perhaps even within a business, you should know how to meet various needs. You do this by answering a few basic questions. Then you will take your answers and convert your requirements to configurations the software uses. After you have tested your configurations, you can move them to your production environment.

How to Convert Your Trading Requirements to Configurations

The answers to the previous questions will help you determine which entities you need to configure in the Workbench. The following table identifies possible configuration entities you will define. Remember that as you become a fluent user, you will find various ways to configure your trading requirements.

Trading Question	Configuration Entities
1. What is the structure of the incoming data?	Standards (to define the structure of the inbound wrapper) Locations (Inbound) (to specify the criteria for identifying the incoming standard)
2. What do you need to do to process the data?	Trade Agreement Profiles (to specify the action to perform; for translation also specifies the outbound wrapper(s), document(s) and map(s).) Acknowledgment Profiles (if required, to specify how to generate the acknowledgment)
3. If you need to translate, what is the standard and version of the outgoing data?	Standards (to define the structure of the inbound document and of the outbound wrapper and document)
4. If you need to translate, how do you create the data?	Maps (to provide instructions to create data for outbound wrappers, documents, and acknowledgments) Trade Agreement Profiles (to specify the outbound maps, and which translations to perform on the input) Acknowledgment Profiles (to specify the maps to use to generate the acknowledgment and wrapper)
5. Will you always know the sender or receiver in advance, and if so, who are they?	Partners, Partner Relationships, Groups (to specify trade agreements, acknowledgments, routing, and special partnership relationships or groups) Locations/StdID (Inbound) (to specify default trade agreements, acknowledgments, partners, or routing options)
6. How do you know where to route the output?	Do you need to route the output to a specific location ID based on a recipient ID or a sender ID? Can you send the output to a generic location?

The following table is a list of the entities that you may want to configure, and the basic purpose of each configuration.

Configuration Entity	Basic Purpose
Standard	Defines the properties of the standard and version and the structure of wrappers, documents, and acknowledgments to parse incoming data and generate outgoing data.
Maps	Provides instructions to create the outgoing data.
Trade Agreement Profiles	Identifies the work to do.
Acknowledgment Profiles	Identifies the requirements for creating a specific acknowledgment.
Locations (Inbound)	Provides information to identify the incoming wrapper as well as perform default processing and default routing.
Partners	Provides user IDs, location IDs, and links to specific trade agreements and acknowledgments.
Partner Relationships	Identifies those partners that trade on a one-to-one basis.
Groups	Identifies those partners that trade within a closed group of predefined partners.

Basic Questions to Determine Trading Requirements

To determine your trading requirements, you must answer some basic questions. The following table gives the questions and provides a brief description of what that question means.

Trading Question	Description
1. What is the standard and version of the incoming data?	You must have a definition to parse the data.
2. What do you need to do to process the data?	Do you need to route a file whose content structure you may not know? Do you need to strip header information and then route a file whose content structure you may not know? Do you need to validate the wrappers, which requires

Trading Question	Description
	<p>scanning the entire structure, and then route the input?</p> <p>Do you need to validate the wrappers and contents, which requires scanning the entire structure, and then route the input?</p> <p>Do you need to translate the input and then route the output?</p>
3. If you need to translate, what is the standard and version of the outgoing data?	You must have a definition to generate the outgoing data.
4. If you need to translate, how do you create the data?	<p>Do you need to create a specific map for a specific partner or can you create a generic map that will be used by several partners?</p> <p>Do you need to process the input document in different ways to create multiple, separate output documents?</p> <p>Do you need to return acknowledgments to the sender?</p>
5. Will you always know the sender or receiver in advance, and if so, who are they?	<p>Will you provide a service for other companies, and not necessarily know the sender or receiver in advance?</p> <p>Will you trade with people or companies you don't know?</p> <p>Will you only trade with partners you know?</p> <p>Will you trade with any partner in a predefined group of partners?</p>
6. How do you know where to route the output?	<p>Do you need to route the output to a specific location ID based on a recipient ID or a sender ID?</p> <p>Can you send the output to a generic location?</p>

Using a Template Approach for New Configurations

To develop a template approach, you should select a few trading partners that represent most of your EDI processing. Then you can develop and test configurations for these partners. This approach requires that there be considerable commonality in the way you trade with the partners.

The result of your initial configuration effort will most likely reveal that you tend to support limited types of trading. Most businesses will be able to implement the use of a template approach for new configurations. A template approach means that you can use a combination of pre-defined configurations and examples of how to combine these with additional configurations to implement new trading partnerships easily.

This system has been designed to help you with this approach, because it separates those configurations that typically do not change from those that do. Therefore, you will notice that the Data Explorer window contains configurations that are relatively stable, such as standards definitions, trade agreement profiles, acknowledgment profiles, and maps. The Partner Explorer, on the other hand, contains those configurations that are more likely to change because they support partnerships, such as inbound locations and partner definitions.

Several utilities will help you use your templates, once you decide what they should be. You can access these utilities from the **File** menu.

File Menu Utility	Description
Import	Allows you to add and optionally rename previously exported definitions to a configuration database.
Export	Allows you to move definitions to an external text file.
Copy to	Allows you to copy renamed definitions to the same database and to copy definitions to other databases, with or without renaming.

Integrating Test and Production Processes

You should develop a strategy to integrate test and production processes. How you do this depends on the size of your workload and how the work is organized. You may have only one computer or you may have many. The basic configurations are:

- Single system that contains test and production database environments.
- Multiple test computers running the Workbench for testing and a separate computer running production.

No matter your size, you must develop a strategy to maximize workflow and minimize problems. You can find further discussion in the *MW Translator Operator Guide and Reference* or the online help for the MW Translator Operator program in the section "Planning and Configuration."

Configuration Task Tree

The following task tree shows one possible order of activities required to complete a configuration. The Workbench allows you to use other orders, because it does not require a definition to be complete to save it. Since some definitions reference other definitions, the following order allows for all cases.

This represents the large picture for configuration tasks. Note that not all tasks are mandatory. The tasks you perform depend on the trading requirements.



For more information about each task refer to the appropriate section.

Determining Processing Control

Overview (Determining Processing Control)

Three mandatory activities determine how you process incoming data as follows:

- Identify the incoming standard
- Find a trade agreement profile
- Find routing instructions (locations)

A failure in any of these will terminate processing. The configurations you create must support these activities. Additional configurations provide optional processing controls.

This chapter discusses the basic issues and strategies you can use to support these mandatory activities. Other chapters discuss individual configurations in more depth.

In order to better understand your strategies, pertinent processing steps describe how the configuration information is actually used.

Identifying the Incoming Standard

The purpose of identifying the incoming standard is to be able to parse the incoming wrapper. The wrapper contains many clues that will determine what to do with the remaining data.

The configurations required here are an input location and standard ID. The Standard ID window also contains default processing definitions for partners, routing, trade agreements, and acknowledgments, which are not used in the standard identification process.

Defining Inbound Locations

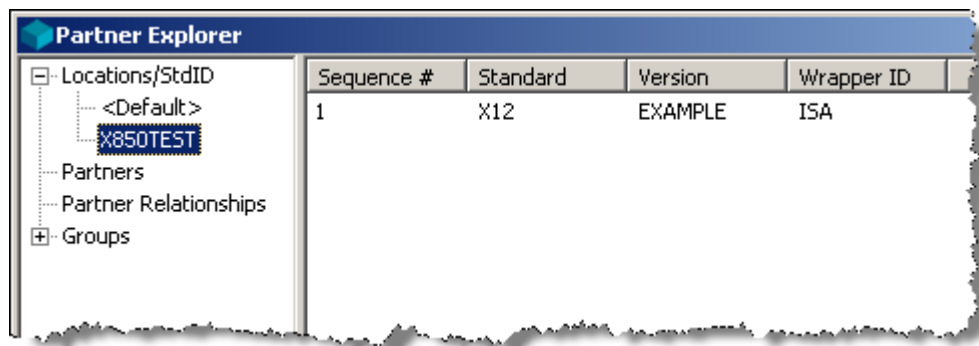
The first step in configuration is to associate the wrapper with an incoming location. There are two types of locations: <Default> and specific or named. Inbound wrappers are associated with one of these types of locations. A MessageWay adapter might specify an inbound location when it delivers data to MessageWay. For more information, refer to the documentation for a specific adapter in the *MessageWay User's Guide and Reference*.

The matching criteria that you specify in the Standard ID window for a location are used to compare with the incoming data. If there is a match, the wrapper definition specified for the location is used to parse the wrapper data.

NOTE: When there is no inbound wrapper data, you will define the wrapper as a null wrapper.

When the messaging system passing the data specifies an inbound location, and the inbound location does not exist in the **Locations/StdID** folder, the <Default> location is searched for a matching wrapper.

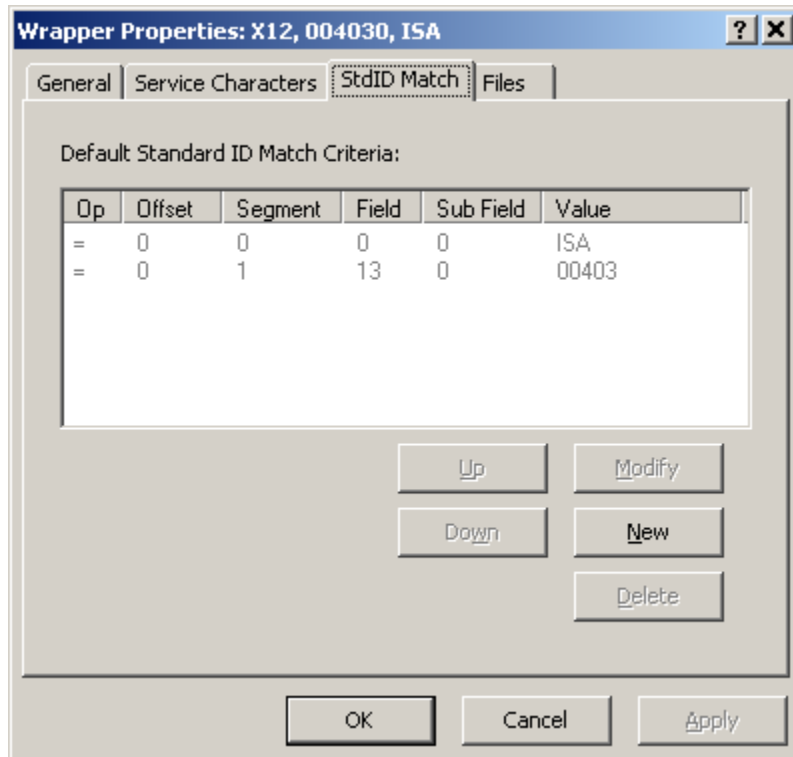
Typically, the <Default> location contains wrapper definitions for the public standard wrappers. You can define all wrappers in the default location if you wish. If you have special processing requirements, you can create a different location for your inbound wrapper. For further discussion on when to use the default location or a specific location, refer to the topic, *Task 6. Specifying the Source Location* (on page 300).



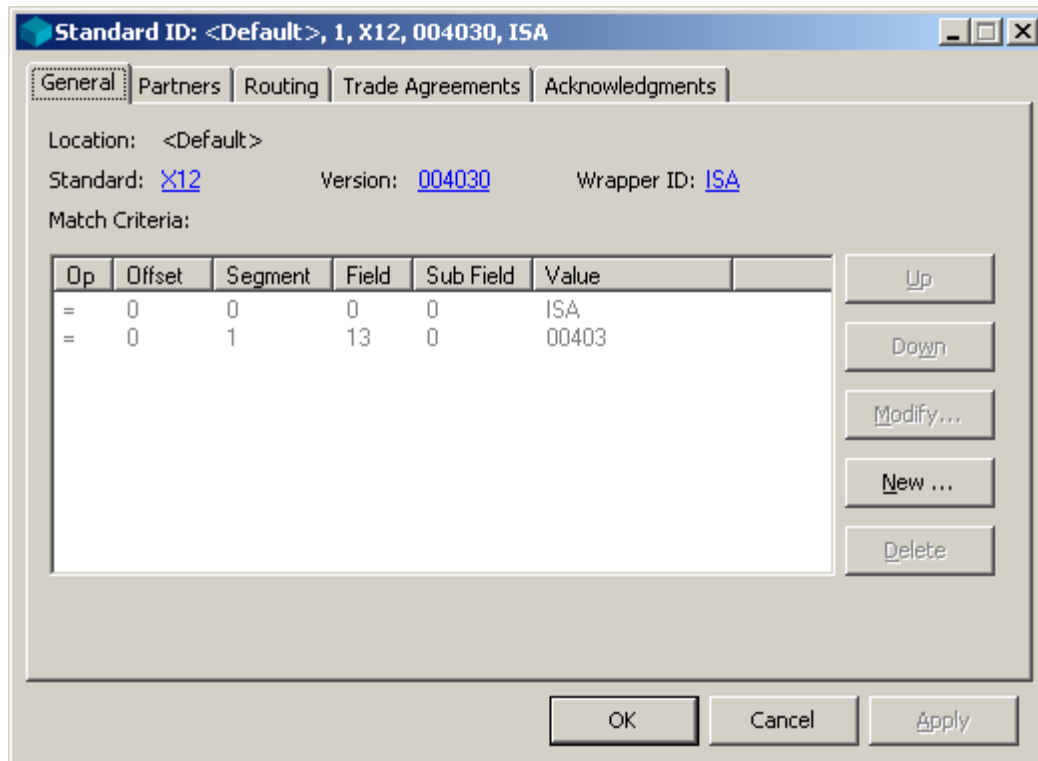
Defining Matching Criteria

In order to select a wrapper definition to parse the inbound wrapper, incoming data must match data specified in the **Match Criteria** list. The matching value must occur within the first block of data, typically 1-4k, depending on the system. For a description of how MW Translator uses the matching criteria, refer to the topic, *Step 1. Identify the Incoming Standard* (on page 47).

When you create a wrapper definition, you have the option of specifying default matching criteria on the **StdID Match** tab of the Wrapper Properties window. The public standard wrappers supplied with MW Translator already have default matching criteria defined. When you select one of these standard wrappers, the matching criteria appear on the **General** tab of the Standard ID window.



When the wrapper is selected for a source location, the criteria are transferred to the **General** tab of the Standard ID window.



Specifying a Trade Agreement Profile

The second mandatory task during processing is to find a trade agreement. The trade agreement specifies the processing tasks. Processing terminates if a trade agreement is not found. Although it is not mandatory, a related requirement is to find an acknowledgment profile, if an acknowledgment is to be generated.

Criteria to Define Senders or Recipients

The question becomes:

- **What inbound data about the sender or recipient do I want to match in order to invoke the trade agreement or acknowledgment profile?**

The type of processing control and implications for each choice are given in the following table, in order from the most restrictive matching criteria to the least restrictive.

Type of Processing Control	Implications
Partner Relationship	This one-to-one type of processing is the most restrictive definition to which you can attach a trade agreement. This requires that both the sender and recipient be defined so the TRM can match on both sets of IDs.
Closed group	This one-to-limited number type of processing is slightly less restrictive than a partner relationship definition. This requires that the sender be defined so the TRM can match on the sender ID, and that the trade agreement be associated with either a recipient partner or the incoming wrapper (standard ID).
Partner	This anyone-to-one type of processing is the most common. It requires that you define the recipient partner so the TRM can match on the recipient ID. You can define the sending partner also if you want to check IDs or specify an acknowledgment, but the recipient can receive data from any sender.
Standard ID (inbound wrapper)	This anyone-to-anyone type of processing is the least restrictive. You do not need to define either a sending or recipient partner since you are not matching on any IDs.

Even though you know what your configuration options are in order to find a trade agreement profile, you have to figure out what questions to ask in order to determine which option you should use. What information about the partners must be matched that will ensure use of the correct trade agreement? The following table gives you questions you should ask and some of the associated configuration implications.

Question	Configuration Implications
Must I define the recipient?	Matching on recipient partner IDs allows you to specify a trade agreement for that partner, alternative IDs for the output wrapper, other characteristics of the output, and a specific destination location.

Question	Configuration Implications
Must I define the sender?	Matching on sender partner IDs allows you to specify an acknowledgment profile for that partner, other characteristics of the output, a specific source location for output, and specific source and destination locations for the acknowledgments created.
If I must define both the sender and the recipient, is there anything about their processing that requires a unique trade agreement?	Matching on sending and recipient partner IDs can be done by creating a partner definition for each. However, if the trade agreement must be unique to their trading relationship, then you must create a partner relationship for these partners. The partner relationship definitions build on the individual partner definitions. It is the most restrictive of all types of processing control.
If I do not have to recognize the sender and/or the recipient, can I use default processing?	Relying only on matching that occurred during the standard identification process allows you to specify a generic trade agreement or acknowledgment profile to process any data that might be sent using this inbound location and wrapper.

Understanding Trade Agreement Profiles

You do not have to define partners to enable and control processing. Trade agreement profiles define what to do. You attach it to a partner, partner relationship or standard ID definition.

Trade Agreement Profiles can provide the following type of information:

- The action you want to perform (route file, route file without header, validate wrappers, validate contents, or translate)
- The inbound document definition, to parse document data
- Inbound security document
- Security user exit
- The response to the document in case of errors (accept, reject)
- For translation, the various output documents you want to create from the same input document
- The maps for the outbound wrapper and document
- Summary documents

The most important question is:

- **To which definition do I attach my trade agreement or acknowledgment profile?**

The answer to this question determines whether you must define partners. You may only need to define the recipient, the sender, or perhaps no partner. This concept may be alien to many EDI users, who have only ever specified processing based on known partners. You can still do this by defining a partner relationship, but this is the most restrictive type of control you can define. It is important that you make a decision regarding how to control processing that keeps your configurations as simple as possible. Reducing the number and complexity of your configurations limits the amount of maintenance you must do and the cost of maintaining them. Simpler is better.

You can define partners for reasons other than to associate it with a trade agreement or acknowledgment profile, such as:

- For security by matching IDs from inbound wrappers
- For alternative routing locations
- To provide alternative partner IDs for outbound wrappers

Deciding What Type of Trade Relationship You Need

This is how you can put all of your definitions to use. You have many ways to define your trade relationships by combining various options. It is important to understand the basic options before you begin to refine them. These basic options in order from the most restrictive to the least restrictive are as follows:

- Partner relationship
- Closed groups
- Partners
- Standard ID (incoming wrappers)

Note that the options of **Closed group** and **Standard ID** do not necessarily require partner definitions. You can find more information about defining partners in the section, *Defining Partners* (on page 243). You can find more information about defining standard ID in the section, *Configuring Standard ID and Defaults* (on page 297).

You determine where to attach a trade agreement or acknowledgment profile based on trading requirements.

Definition Using Trade Agreement	Trading Requirements
Partner Relationship	Special processing for sending and recipient partner; any unique information specified in a trade agreement profile that differs from a more generic profile that could be associated with the recipient partner, such as, a unique map, special user exits, summary document.
Partners	Processing for a sending or recipient partner, without regard to the other partners involved; trade agreements and acknowledgment profiles of this type tend to be more generic, and might even be used by many different partners.
Standard ID (incoming wrappers)	Generic processing for any sending or recipient partner. This is often called default processing, because it specifies the only definitions that will allow processing to continue when other more specific definitions did not provide a match.
Groups, Partners, Standard ID	Sender-based processing for a group of sending partners; trade agreement must belong to the same group and can be associated with either a recipient partner or an incoming wrapper.

Processing Step - Looking for Defined Partner IDs

Though the TRM can do without partner profiles, it checks for valid partner IDs at this point. For a description of this step, refer to the topic, *Step 4. Look for Defined Partner IDs for the Sender and Recipient at this Level* (on page 53).

Processing Step - Finding Partner Profiles for the Sender and Recipient

For a description of this step, refer to the topic, *Step 5. Find Partner Profiles for the Sender and Recipient Based on Partner IDs* (on page 55).

Processing Step - Finding a Partner Relationship Definition

For a description of this step, refer to the topic, *Step 6. Find a Partner Relationship Definition, If One Exists* (on page 57).

Processing Step - Repeating Preceding Steps for Remaining Wrapper Levels

For a description of this step, refer to the topic, *Step 7. Repeat Steps 3-6 for Remaining Wrapper Levels 2 through 4, if They Exist* (on page 57).

Specifying Routing Locations

The third mandatory task is to find locations to route the data after processing. Any data generated during processing must be routed to a location. The MessageWay messaging system will then deliver the data based on the destination location. MessageWay also requires that a source location be identified. Both source and destination locations must be found in the configurations or processing terminates.

NOTE: The source location is usually associated with a sending partner. If there is no sending partner identified, then it may be specified by default routing in the standard ID record used during parsing.

You specify locations for output differently than locations for acknowledgments.

Locations for Output

Output locations can be found on any of several configurations and may be specified on a **Conditions** tab of a Map window using an Edibasic statement.

Routing locations might be determined in two stages, as follows:

- 1 Determine source and destination locations for output using configurations.
- 2 Execute **Condition Method** during mapping, possibly changing those locations.

The following table specifies the strategies used to find outbound locations using configuration information, and the order in which they are searched, beginning with the definition on which the trade agreement was found.

IMPORTANT: An Edibasic instruction in a document-level **Condition** method in the map will override locations determined according to the strategies in the following table.

Order	Strategy for Determining Location ID from Configurations
1	Use alternative locations on Trade Agreement Options window for sending or recipient partner

Order	Strategy for Determining Location ID from Configurations
2	Use aliases, if alias found on trade agreement properties
3	Use locations defined for inbound partner relationship or sending or recipient partner
4	Use routing defined for Standard ID window
5	Override output source location with inbound location used during standard identification if Use Source Before Partner Routing is checked

Processing Step - Finding Routing Locations for Translation

The following information describes in detail the steps taken during processing to search for source and destination locations for the outbound interchange. There are two processing options to specify a location: configurations and mapping. You can override a location found using the configurations by mapping one in a condition statement. If no location is found, processing terminates.

Determine Location Using Configurations

The TRM first attempts to determine the location using the configurations. You can provide an alternative location for the recipient partner directly on the Trade Agreement Options window. Alternatively, you might use an alias type on the Trade Agreement Profile. For a description of this step, refer to the topic, *Step 14. Determine Routing (Translation)* (on page 91).

Override Configured Location Using Mapping

The TRM will use the location specified during mapping, overriding any location determined using the configurations.

NOTE: If no source or destination location is found, processing stops.

Processing Step - Finding Routing Locations for Validation

For a description of this step, refer to the topic, *Step 15. Determine Routing (Validation)* (on page 91).

Locations for Backward Acknowledgments

Output locations can be found on any one of several configurations. The first location found is the one that is used. Note that unlike trade agreement routing, there is no use of aliases for acknowledgment routing. The following table specifies the strategies used to find locations for acknowledgments and the order in which they are searched, beginning with the definition on which the acknowledgment was found.

Order	Configuration Strategy for Location ID
1	Use alternative locations on the Acknowledgments tab of the Acknowledgments Options window.
2	Use locations defined for inbound partner relationship or sending or recipient partners.
3	Use routing on the Routing tab of the Standard ID window.
4	Override acknowledgment destination location with inbound location used during standard identification if Use Source Before Partner Routing box is checked.
5	If source or destination location is not found, terminate processing.

Processing Step - Finding Routing Locations for Backward Acknowledgments

For a description of this step, refer to the topic, *Step 19. Determine Routing for Backward Acknowledgments* (on page 103).

This page intentionally blank

Quick Configuration Tour and Test

Overview

The Translator Runtime module (TRM) must perform at least 3 tasks in order to successfully process the incoming data. These tasks are:

- 1 Parse the incoming data
- 2 Determine what work must be done
- 3 Determine where to send the output

Processing Flow

The following table shows you what must happen to complete the listed tasks. For a more detailed explanation of this process, refer to the topic, *Understanding Processing Flow* (on page 45).

Task	Steps
Parse incoming data	<ol style="list-style-type: none"> 1 Match raw data with inbound wrapper information to find wrapper definition. 2 Close file and reopen, using wrapper definition to parse wrapper and extract data.
Determine what work must be done	<ol style="list-style-type: none"> 1 Match parsed wrapper data to find a trade agreement profile, which might be attached to 1 of 3 configurations: <ul style="list-style-type: none"> ▪ Partner Relationship ▪ Partner ▪ Standard ID 2 Use trade agreement profile to determine what to do: <ul style="list-style-type: none"> ▪ Route File ▪ Route File w/o Header ▪ Validate Wrapper ▪ Validate Contents ▪ Translate 3 Find additional configurations to accomplish work item listed in #2: <ul style="list-style-type: none"> ▪ Validate wrapper: no other definitions required ▪ Validate contents: find document definition to parse and validate document ▪ Translate: find wrapper and document definitions and wrapper and document maps for all outputs to translate document 4 Search list for acknowledgments that match incoming wrapper and create up to 9 acknowledgments on list, which might be associated with 1 of 3 configurations: <ul style="list-style-type: none"> ▪ Partner Relationship ▪ Partner ▪ Standard ID

Task	Steps
Determine where to send the output	<ul style="list-style-type: none">▪ Find a location that might be associated with 1 of 4 configurations:<ul style="list-style-type: none">▪ Partner Relationship▪ Partner▪ Standard ID▪ Edibasic Mapping instruction

Processing Options

Which definitions are required depends on the work. There are five options, each of which may require a responding acknowledgment:

- 1 Route input file after matching header
- 2 Route file without a header, using defaults
- 3 Validate the wrapper(s) and route the input
- 4 Validate the wrapper(s) and content and then route the input
- 5 Translate the contents, which includes creating and routing new output file(s) as well as validating the wrapper(s) and contents

You will need to provide different configuration information, depending on which of the options you choose and how you specify to find the information.

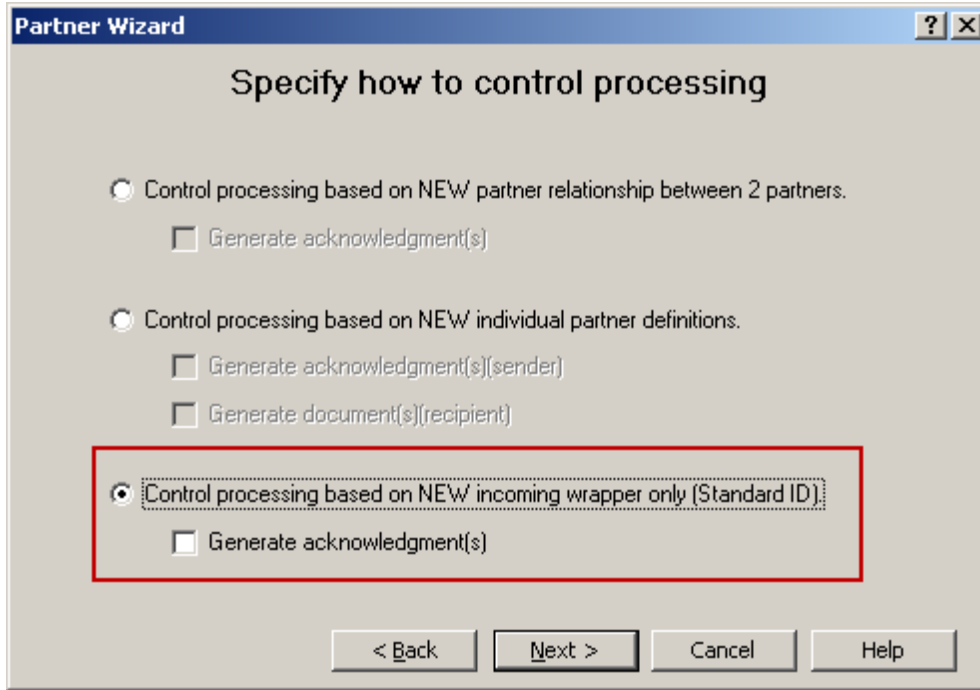
You can accomplish most tasks with or without partners, so we will choose the option for each task that would be a typical configuration. You can use these tables to develop your own configuration templates.

Validate the Wrapper and Route the Input

When you validate only wrapper format and data, you provide a service that is typical of EDI value-added networks (VANs).

For purposes of this discussion, you will not validate partner IDs. This is the typical choice if you do wrapper validation and routing and do not need to verify incoming partner IDs. This scenario is often called *open trade*. What it means is that you would associate a trade agreement profile with an incoming standard ID wrapper rather than a partner or partner relationship.

When you use the Partner Wizard to create trade relationships, you notice that one of the selections is **Control Processing based on NEW incoming wrapper only (Standard ID)**.



Partner Wizard Window

Partner definitions are used if they exist. If partner definitions match the IDs in the incoming data, an attempt is made to find a matching trade agreement profile associated with the partner.

For this discussion, we do not want to use trade agreements associated with a partner ID. To make sure a trade agreement is not found on a matching partner record, and thus use the trade agreement profile from Standard ID, you have a couple of options. The easiest to configure is to make sure that there are no partner records that use the incoming sender or recipient partner IDs.

The following table indicates the minimum tasks and the related configurations that you need to supply for this scenario.

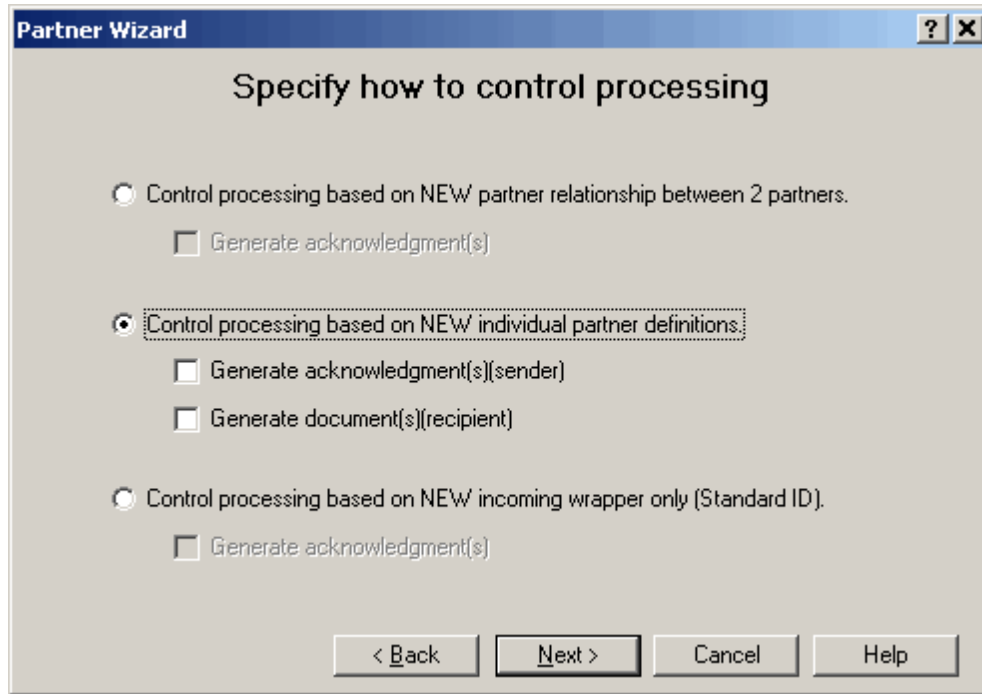
Basic Task	Configuration Information (location)
Identify incoming wrapper (search source location for matching wrapper)	<ul style="list-style-type: none"> ▪ Inbound location (Partner Explorer, folder) ▪ Input wrapper definition (Wrapper, Segment, Element windows) ▪ Input wrapper specified (Standard ID window, General page)
Find Partners (search partner definitions for	<ul style="list-style-type: none"> ▪ No partners defined Partners folder, Partner Relationships folder)

Basic Task	Configuration Information (location)
matching partner IDs)	<ul style="list-style-type: none"> ▪ Partner definitions NOT required (Standard ID window, Partners page,)
Determine what to do (search for matching trade agreement profile and find action, no document matching required)	<ul style="list-style-type: none"> ▪ Trade Agreement Profile window ▪ Trade agreement specified (Standard ID window, Trade Agreements page, Trade Agreement Profile) ▪ Wrapper information (standard, version, release, agency) for matching (Trade Agreement Profile window, Matching Fields page) ▪ Action Validate Wrappers (Trade Agreement Profile, Options page)
Find outbound location	<ul style="list-style-type: none"> ▪ Location (Standard ID window, Routing page)
Create optional Acknowledgment (search for matching acknowledgment profile, determine conditions of creation)	<ul style="list-style-type: none"> ▪ Outbound acknowledgment definition (Document, Segment, Element windows) ▪ Outbound acknowledgment wrapper (Document, Segment, Element windows) ▪ Acknowledgment map (Map window) ▪ Acknowledgment Profile window ▪ Acknowledgment information (Standard ID window, Acknowledgments page, Acknowledgment Profile window) ▪ Create Acknowledgment rules (Acknowledgment Profile, Options page)
Find acknowledgment location	<ul style="list-style-type: none"> ▪ Default Acknowledgment Routing (Standard ID, Routing page)

Validate Wrapper and Contents and Route the Input

When you validate contents as well as wrappers, you may want to do so based on a partner implementation or not. We have chosen to use partners here to illustrate the difference between partner-based routing and Standard ID routing. Furthermore, let us assume that we want to use only the recipient partner definition, and let the acknowledgment definition default to Standard ID, as in the previous example. Therefore, we want to make sure we define only the recipient partner, not the sending partner. In addition, for content validation, the incoming document must also be parsed, not just the wrappers.

When you use the Partner Wizard to create trade relationships, you notice that one of the selections is **Control Processing based on NEW Individual Partner Definitions**.



Partner Wizard Window

The following table indicates the minimum tasks and the related configurations that you need to supply for this scenario.

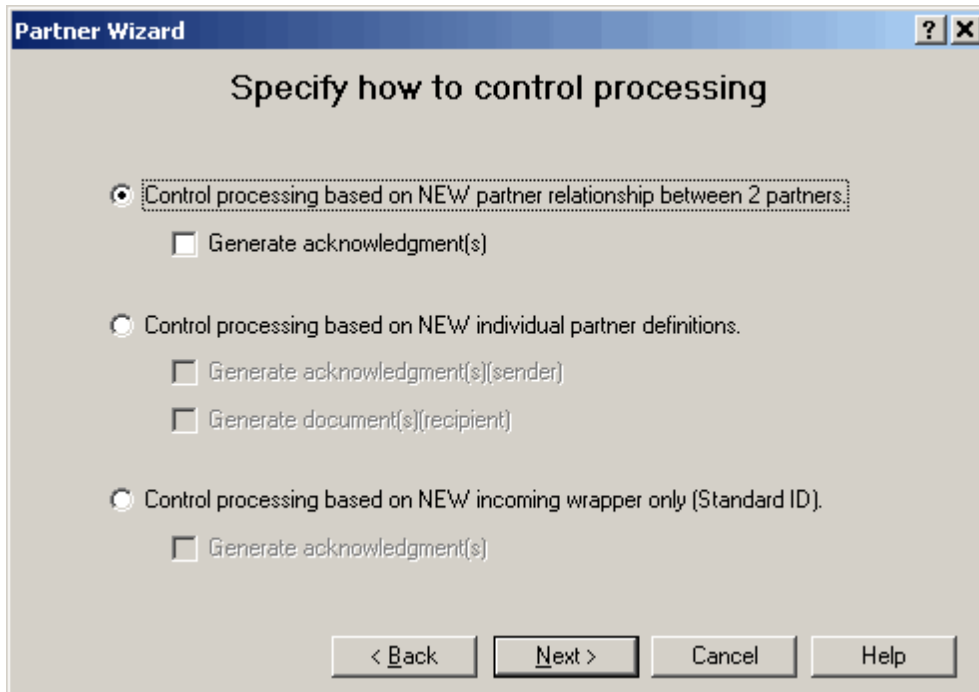
Basic Task	Configuration Information (location)
Identify incoming wrapper (search source location for matching wrapper)	<ul style="list-style-type: none"> ▪ Input wrapper definition (Wrapper, Segment, Element windows) ▪ Input wrapper specified (Standard ID, General page)
Find Partner (search partner definitions to match recipient IDs, use default routing for sender's acknowledgment)	<ul style="list-style-type: none"> ▪ Recipient Partner (Partner, IDs page) ▪ Sending Partner definition not required (Standard ID, Partners page)
Determine what work to do (search for matching trade agreement profile and find action, document matching required)	<ul style="list-style-type: none"> ▪ trade agreement (start with matching partner relationship or partner definition, Trade Agreements page, Trade Agreement Profile) ▪ wrapper information (standard, document ID, version, release, agency) that matches inbound data parsed from wrapper (Trade Agreement Profile, Matching Fields page) ▪ document definition (Document, Segment, Element windows) ▪ document specified (Trade Agreement Profile, Input page) ▪ action Validate Contents (Trade Agreement Profile, Options page)
Find outbound location	<ul style="list-style-type: none"> ▪ Location (first check Trade Agreement Properties, Recipient Partner page, then if not found, check Partner, General page)
Create optional Acknowledgment (search for matching acknowledgment profile, determine conditions of creation)	<ul style="list-style-type: none"> ▪ acknowledgment (Standard ID, Acknowledgments page, Acknowledgment Profile) ▪ Create Acknowledgment (Acknowledgment Profile, Options page)
Find acknowledgment location	<ul style="list-style-type: none"> ▪ Default Acknowledgment Routing (Standard ID, Routing page)

Translate

You can do translation without defining partners, in which case you would need to make sure that a trade agreement profile could be found somewhere else.

However, in order to illustrate a partner relationship, we assume here that you want to validate both the sender and recipient IDs. In addition to what you have seen before, you will now have another partner to define, an outbound standard, and maps for the outbound documents and wrapper.

When you use the Partner Wizard to create trade relationships, you notice that one of the selections is **Control Processing based on NEW partner relationship between 2 partners.**



Partner Wizard Window

The following table indicates the minimum tasks and the related configurations that you need to supply for this scenario. Note that this scenario is similar to the one used for the EXAMPLE.

Basic Task	Configuration Information (location)
Identify incoming wrapper (search source location for matching)	<ul style="list-style-type: none"> ▪ Inbound location (Partner Explorer, Locations/StdID folder) ▪ Inbound wrapper definition (Wrapper, Segment, Element)

Basic Task	Configuration Information (location)
wrapper)	windows) <ul style="list-style-type: none"> ▪ Inbound wrapper specified (Standard ID, General page)
Find partner (search partner relationship definitions to match sender and recipient IDs)	<ul style="list-style-type: none"> ▪ Partner relationship for sending and recipient partners (Partner relationship, Partner, IDs page)
Determine what work to do (search for matching trade agreement profile and find action, document matching required)	<ul style="list-style-type: none"> ▪ trade agreement (start with matching partner relationship, Trade Agreements page, Trade Agreement Profile) ▪ wrapper information (standard, document ID, version, release, agency) that matches inbound data parsed from wrapper (Trade Agreement Profile, Matching Fields page) ▪ document definition (Document, Segment, Element windows) ▪ document specified (Trade Agreement Profile, Input page) ▪ action Translate (Trade Agreement Profile, Options page)
Translate	<ul style="list-style-type: none"> ▪ For each output (Trade Agreement Profile, Outputs page) ▪ document definition (Document, Segment, Element windows) ▪ document map ▪ wrapper definition (Wrapper, Segment, Element windows) ▪ wrapper map
Find outbound location	<ul style="list-style-type: none"> ▪ Location (first check Trade Agreement Properties, Recipient Partner page, then if not found, check Partner, General page)
Create optional acknowledgment (search for matching acknowledgment profile, determine conditions of creation)	<ul style="list-style-type: none"> ▪ acknowledgment (Standard ID, Acknowledgments page, Acknowledgment Profile) ▪ Create Acknowledgment (Acknowledgment Profile, Options page)
Find acknowledgment location	<ul style="list-style-type: none"> ▪ Default Acknowledgment Routing (Standard ID, Routing page)

Although there are many ways to configure a trade relationship, you will soon know which definitions you must provide to meet 80 percent of your needs. For the remaining 20 percent of your relationship needs, you have the tools to meet those as well.



Reviewing Sample Configurations

To understand configurations, reviewing an example is helpful. Although this provides a limited view of what you can do with configurations, you do have access to a working example. If someone has not already done so as part of the software installation process, you can load this example from the /EXAMPLES directory using the Workbench. Before you proceed, you must also understand how to access multiple database configuration environments.

Checking Your Environment

Since you can have multiple databases, which means multiple environments, you must know which environment you are currently accessing. You make the Workbench aware of the environment by adding it to the environment list. For instructions about how to add environments to the environment list, refer to the topic, *Select Environment Command* (on page 525).

Check your environment to make sure you are using the correct database by looking at the path listed in the title bar of the Workbench.

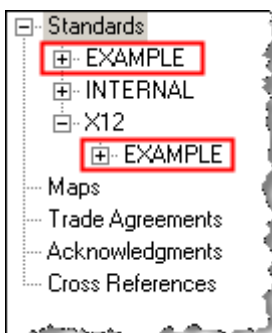
You can change environments by selecting the appropriate one from the list  associated with the **Select Environment** button 

Checking for the Example Standard

Before proceeding, let us check to see if the test example definitions exist in the Workbench.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, double-click **Standards**.
- 2 Make sure that a folder called **EXAMPLE** exists both under the **Standards** folder and under the **X12** folder. You need the definitions under the X12 folder for this exercise.



If you see both **EXAMPLE** folders, you can assume the example files were imported, so proceed to the topic, Testing the Example.

If you do not see both **EXAMPLE** folders, the assumption is that someone deleted the definitions, because they are always loaded when you install a copy of the Workbench.

- 3 From the **File** menu, choose **Import**.

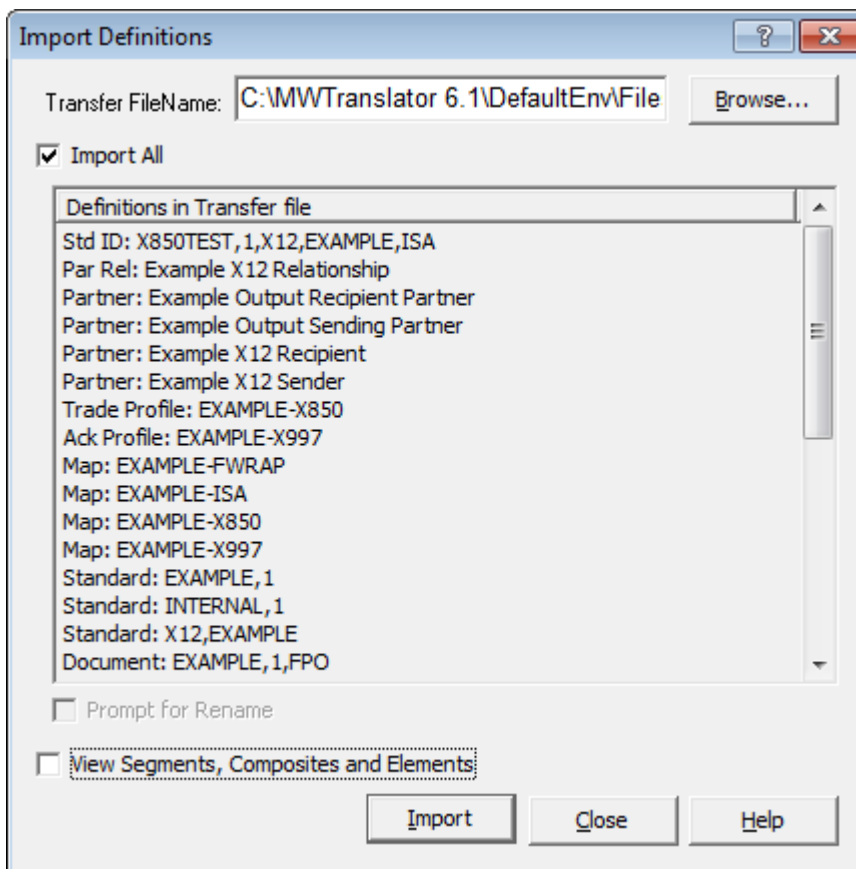
The **Import Definitions From** dialog box appears.

- 4 Navigate to the **X850Test** folder within the *<Install Directory>*\MW Translator 6.2\Workbench\Examples folder.

- 5 Select the **X850test.trn** file, and click **Open**.

The **Import Definitions From** dialog box appears with all items on the list selected.

- 6 Select the **Import** button.



- 7 If for some reason any of the definitions are already in your database, you will get a message asking if it is OK to replace the duplicate definitions with those from the file. To overlay all without further questions, select **YestoAll**.

- 8 Press **F5** to update the lists on the Data Explorer window. Check to make sure the standard you need was imported by repeating steps 1 and 2. If you do not see the **EXAMPLE** folder within the **X12** folder, you should repeat this process.

- 9 If necessary, also from the **X850Test** folder, copy the input file X850test.txt to **C:\MWTranslator 6.2\DefaultEnv\In**.

Configurations for EXAMPLE Test

We have included all of the definitions and files to support a sample translation of an X12 850 to a proprietary purchase order, which also creates a 997 acknowledgment. This translation uses a partner relationship to control processing.

Review the following database definitions, which you can find within the specified folders:


Configurations	folder, Definitions
Inbound location	▪ Locations/StdID, X850TEST
Inbound wrapper	▪ Standards, Versions, Wrappers X12, EXAMPLE, ISA
Partner relationship for sending and recipient partners	▪ Partner Relationships, Example X12 Relationship Partners; Example X12 Sender, Example X12 Recipient
Trade agreement profile	▪ Trade Agreements, Example-X850
Inbound document	▪ Standards, Versions, Documents, X12, EXAMPLE, 850
Outbound document	▪ Standards, Versions, Documents, EXAMPLE, 1, FPO
Outbound document map	▪ Maps, EXAMPLE-X850
Outbound wrapper	▪ Standards, Versions, Wrappers, EXAMPLE, 1, FWRAP
Outbound wrapper map	▪ Maps, EXAMPLE-FWRAP
Acknowledgment profile	▪ Acknowledgments, EXAMPLE-X997
Acknowledgment definition	▪ Standards, Versions, Documents, X12, EXAMPLE, 997
Acknowledgment wrapper	▪ Standards, Versions, Wrappers, X12, EXAMPLE, ISA
Acknowledgment map	▪ Maps, EXAMPLE-X997

Testing the EXAMPLE Translation

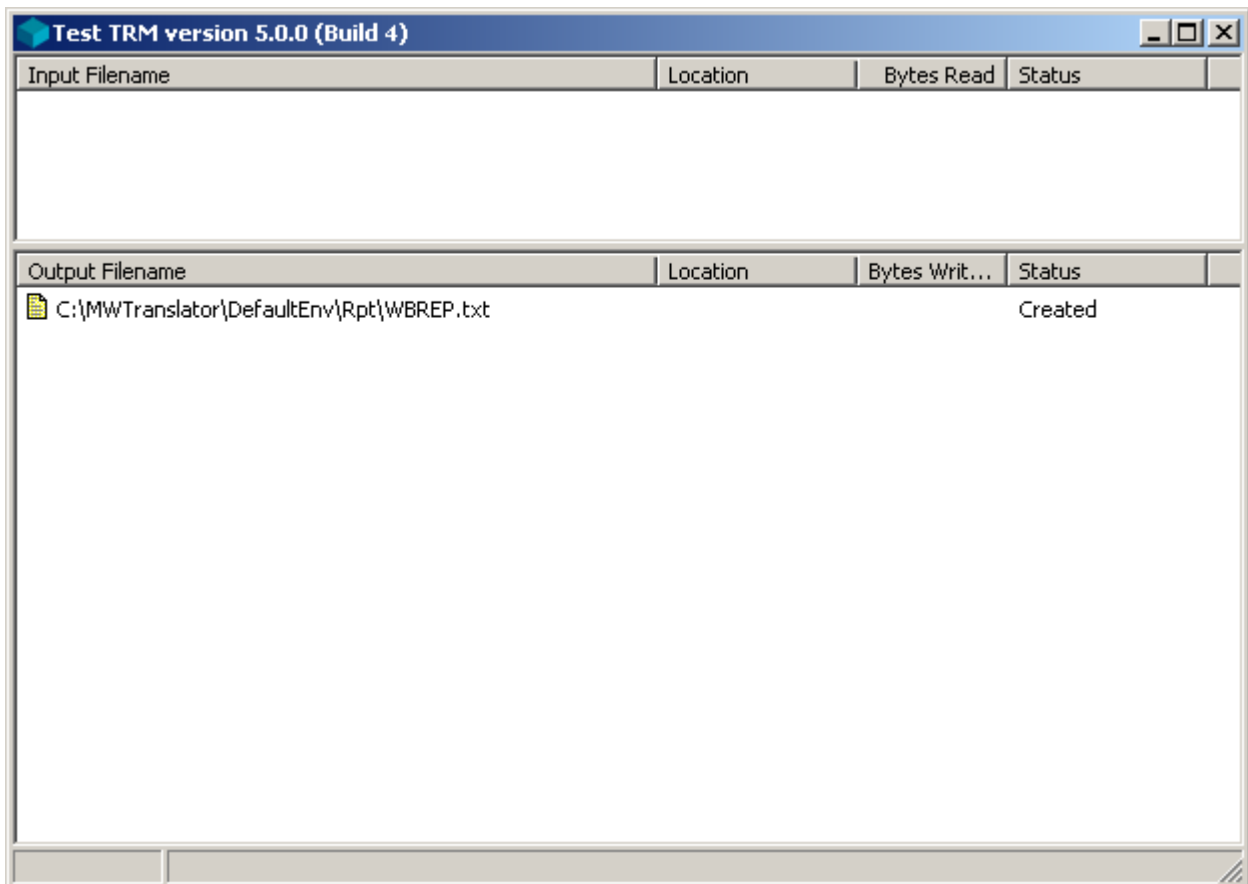
You test your configurations using the Test window.

Running the Example Test

You can test the EXAMPLE translation as follows:

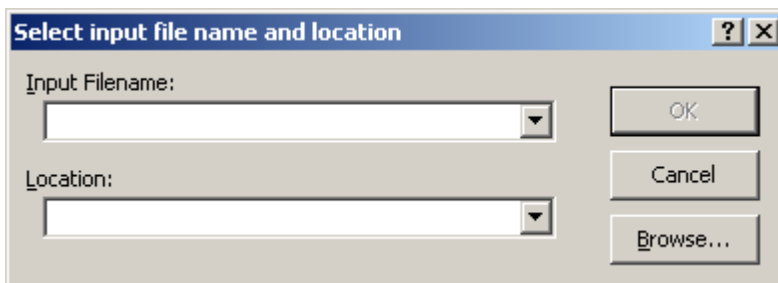
- 1 From the Workbench toolbar, select the **View Test Setup** button .

The Test window appears.

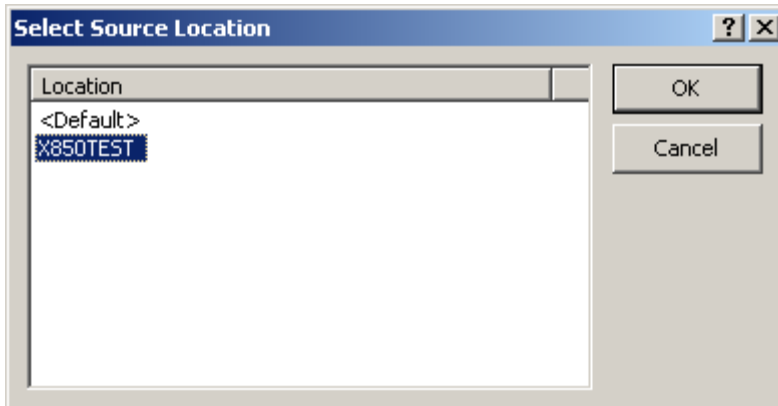


- 2 Select the **Add Test Input** button .

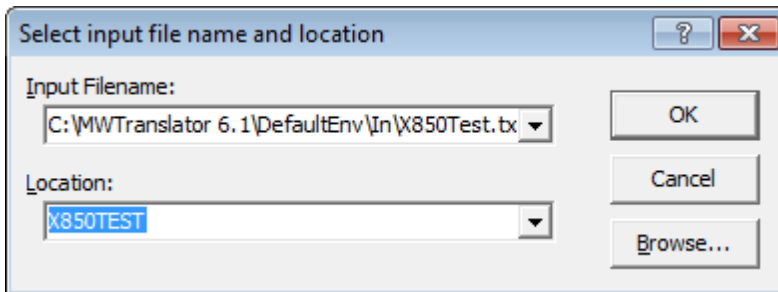
The **Select input file name and location** dialog box appears.




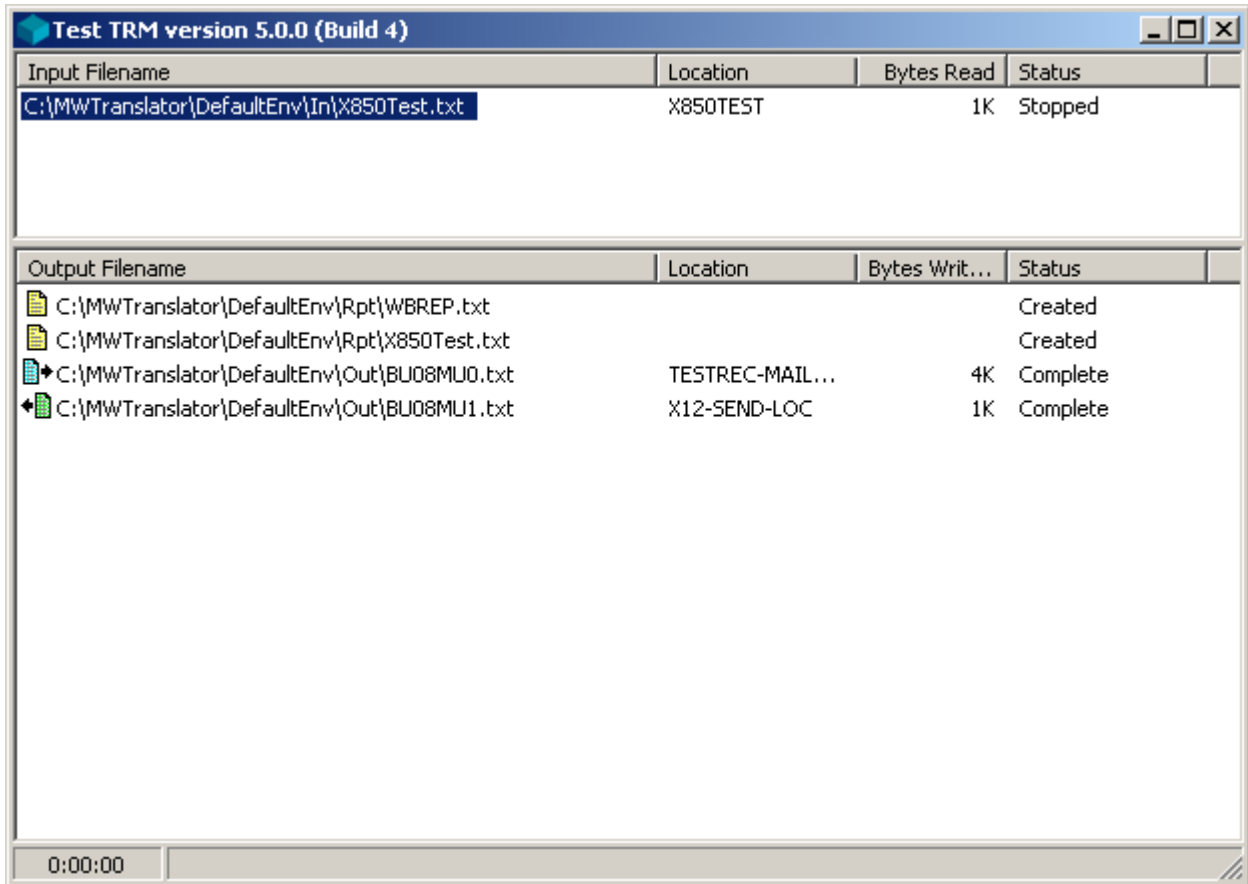
- 3 Place your cursor in the **Input Filename** box, and use the **Browse** button to find the input file **x850test.txt**.
- 4 Place your cursor in the **Location** box, and then use the **Browse** button to find your inbound location, **X850TEST**, and select **OK**.



- 5 Your window should appear as follows. Select **OK**.



- 6 Make sure your input file is selected, and select the **Run Test** button . After processing completes, your window will look as follows:




The screenshot shows a window titled "Test TRM version 5.0.0 (Build 4)". It contains two tables. The first table, "Input Filename", has one row: "C:\MWTranslator\DefaultEnv\In\X850Test.txt" with Location "X850TEST", Bytes Read "1K", and Status "Stopped". The second table, "Output Filename", has four rows: "C:\MWTranslator\DefaultEnv\Rpt\WBREP.txt" (Created), "C:\MWTranslator\DefaultEnv\Rpt\X850Test.txt" (Created), "C:\MWTranslator\DefaultEnv\Out\BU08MU0.txt" (Complete, 4K, Location TESTREC-MAIL...), and "C:\MWTranslator\DefaultEnv\Out\BU08MU1.txt" (Complete, 1K, Location X12-SEND-LOC). A timer at the bottom left shows "0:00:00".



Input Filename	Location	Bytes Read	Status
C:\MWTranslator\DefaultEnv\In\X850Test.txt	X850TEST	1K	Stopped

Output Filename	Location	Bytes Writ...	Status
C:\MWTranslator\DefaultEnv\Rpt\WBREP.txt			Created
C:\MWTranslator\DefaultEnv\Rpt\X850Test.txt			Created
C:\MWTranslator\DefaultEnv\Out\BU08MU0.txt	TESTREC-MAIL...	4K	Complete
C:\MWTranslator\DefaultEnv\Out\BU08MU1.txt	X12-SEND-LOC	1K	Complete

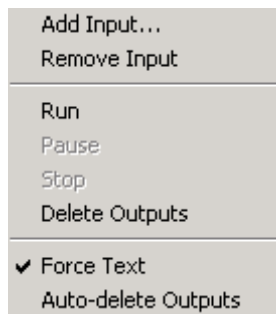
Viewing the Reports and Output

You can double-click any of the input or output files to immediately view the input, reports, or output in a text editor.

There are two reports produced, which are indicated by the report icon, . The first is a configuration report that will display pre-processing errors. The second is the processing report. The processing report file names are reused, because they are created from the name of the input file. If you want to save a particular processing report, rename the existing one before you rerun the test.

For this example there are two output files: an outbound purchase order and a backward acknowledgment. The purchase order output file is indicated by the output icon, . The acknowledgment file is indicated by the backward acknowledgment icon, . The output and acknowledgment file names are unique, because they are created by the system based on a date/time stamp. However, you may want to copy these down or rename them, if you want to quickly relocate a particular output file.

Two options on the **Test** menu will affect your output: **Force Text** and **Auto-Delete Outputs**. The **Force Text** option will create the output in a format that is easier to read by adding carriage return and linefeeds to the end of each segment. The **Auto-Delete Outputs** will delete any files, other than reports, listed on the Output list when you close the Test window. This is useful if you want to keep your system from being cluttered with multiple output files from tests you have run.



Understanding Processing Flow

Overview

The primary task of this software is to deliver EDI services promptly and appropriately to meet the requirements of your trading partnerships. Delivering services promptly is inherent in its design. To deliver services appropriately, the Translator Runtime Module (TRM) relies on configurations and standards definitions. Definitions and maps for acknowledgments and wrappers for all supported public standards are included as part of the installation. You must supply required definitions and maps for your proprietary standard(s). You must also define all trading relationships.

In order to successfully process information, the TRM must perform at least 3 tasks (1, 2, and 4), and optionally a fourth task (3) as follows:

- 1** First, it must find the appropriate wrapper definition for the incoming data to parse the wrapper. This stage is called standard identification.
- 2** Second, it must find a trade agreement profile to be able to determine what it is supposed to do with the data.
- 3** Third, if you want to create a control document or backward acknowledgment, it must also find an acknowledgment profile.
- 4** Fourth, it must find one or more destination locations to which it can direct the output and any acknowledgements.

Whatever else the TRM may do in addition to tasks 1, 2, and 4, and optionally 3, it must, at a minimum, perform these tasks. It cannot continue if it does not succeed with these tasks.

You have many options to control processing with your configurations. To fully understand what you are able to do and how you can best do it, you should understand how the TRM processes incoming data.

How the TRM Processes Documents

Once you understand how the TRM does its work, you will find it is relatively simple to create proper configurations. When you know what the basic tasks are, you can review the specific discussion for each task for more details. This discussion contains the following types of information:

- List of basic tasks in the order performed
- Description of each task
- Windows that display the configurations related to the task
- Areas on a processing report related to the task

The tasks the TRM performs depend on whether it is doing only wrapper validation, contents validation or translation. The interchange is the unit of work. The TRM repeats its list of tasks for each interchange.

The TRM performs its basic tasks on each interchange as follows:

- 1 *Identify the incoming standard* (on page 47)
- 2 *Execute a pre-processing user exit, as required* (on page 49)
- 3 *Parse and validate one level of the incoming wrapper* (on page 50)
- 4 *Look for defined partner IDs for the sender and recipient at this level* (on page 53)
- 5 *Find partner profiles for the sender and recipient based on partner IDs* (on page 55)
- 6 *Find a partner relationship definition, if one exists* (on page 57)
- 7 *Repeat steps 3-6 for remaining wrapper levels 2 through 4, if they exist* (on page 57)
- 8 *Find the acknowledgment profiles, if they exist* (on page 57)
- 9 *Find a trade agreement profile after scanning the document header* (on page 61)
- 10 *Parse the incoming document(s) (contents validation and translation)* (on page 68)
- 11 *Generate outbound documents* (on page 70)
- 12 *Repeat steps 9-11 for remaining documents* (on page 74)
- 13 *Determine IDs for outbound partners* (on page 74)
- 14 *Generate (Validation Only or Translation)* (on page 86)
- 15 *Determine Routing (Validation)* (on page 91)
- 16 *Determine Routing (Translation)* (on page 91)
- 17 *Repeat steps 14 and 16 for each defined output (translation)* (on page 97)
- 18 *Generate backward acknowledgments, as required* (on page 99)
- 19 *Determine routing for backward acknowledgments, as required* (on page 103)

Step 1. Identify the Incoming Standard

The first task is to identify the incoming standard. This is called the *standard identification* process. If the TRM cannot identify the standard, it terminates processing of the message stream. If the TRM does identify the standard, it will then use this information to locate the wrapper definitions of the standard so it can begin parsing the wrapper. The TRM can identify the incoming standard based on source location, data content of the wrapper, or a combination of location and data content of the wrapper.

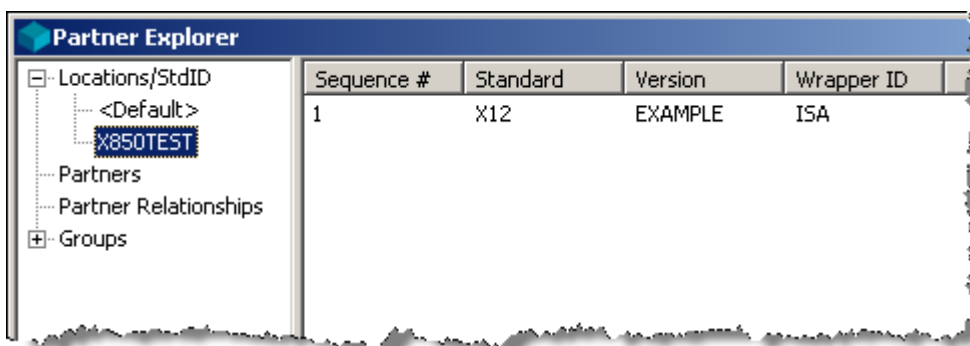
The processing report specifies the version of the TRM, when processing started, and the name of the input file.

```
Test Runtime Module Version 5.0.0(Build 4)
Start of translation 2010/02/16 12:05:20
Source location: X850TEST Input name: C:\MWTranslator\DefaultEnv\In\X850Test.txt
```

IMPORTANT: To identify the incoming standard, the TRM reads the input file in raw mode as binary data. When it identifies the standard, it then re-opens the file and parses the data using the wrapper configuration that it found.

To identify the incoming standard, the TRM uses the raw data as follows:

- 1 Search a location in the Locations/StdID (Partner Explorer) folder for a matching wrapper:
 - Use the source location, which is usually supplied by messaging systems or by you during testing from the Workbench.
 - otherwise –
 - If the location does not exist (location not specified by the adapter or the one specified does not exist), MW Translator searches the list in the <Default> location.



- 2 MW Translator tries to match the inbound data with the matching criteria defined for each wrapper, going in order by sequence number, until it finds a match. If one of the wrappers has no matching criteria specified, and none of the previous wrappers match, it uses that wrapper definition.

IMPORTANT: Any wrapper definition without matching criteria automatically matches, and any wrapper definitions following it will never be used.

Standard ID: X850TEST, 1, X12, EXAMPLE, ISA

General Partners Routing Trade Agreements Acknowledgments

Location: X850TEST
 Standard: X12 Version: EXAMPLE Wrapper ID: ISA

Match Criteria:

Op	Offset	Segment	Field	Sub Field	Value
=	0	0	0	0	ISA

Up
Down
Modify...
New ...
Delete

OK Cancel Apply

IMPORTANT: If the TRM cannot identify the incoming standard, it terminates processing.

The processing report shows the following:

- Source location (X850TEST)
- Numeric sequence of the definition in the location (1)
- Standard (standard, version, wrapper ID) it has identified as a match (X12, Example, ISA)

Identified standard: X850TEST, 1, X12, EXAMPLE, ISA

Step 2. Execute a Pre-processing User Exit, as required.

You may optionally specify a pre-processing method on the Wrapper Properties window. This tells the TRM to execute that pre-processing user exit, which it does after the TRM has re-opened the file.

IMPORTANT: Once the TRM determines the definition of the first incoming wrapper from the standard identification process, it re-opens the input based on the standard and wrapper definitions. When the **IO Mode** is set to **Text**, the TRM replaces the input carriage-return/linefeed (CRLF), linefeed (LF) or record break with a single new-line (NL) character for this and any subsequent interchanges in this file. Therefore, when reading the data with a pre-processing user exit, you must allow for the change in segment terminators to one (NL) character.

Step 3. Parse and Validate One Level of the Incoming Wrapper

The category specifies the parsing routine, such as delimited, fixed, XML, SWIFT. The IO Mode, text or binary, specifies how the TRM determines the end-of-segment for fixed-length data.

The TRM executes Edibasic user validation routines at this point. It parses the data in the wrappers and performs compliance checking. It performs this step one level at a time.

IMPORTANT: Once the TRM determines the definition of the first incoming wrapper from the standard identification process, it re-opens the input based on the standard and wrapper definitions. When the IO Mode is set to **Text**, the TRM replaces the input carriage-return/linefeed (CRLF), linefeed (LF) or record break with a single new-line (NL) character for this and any subsequent interchanges in this file.

Therefore, when you have multiple interchanges in a file where the IO Mode is defined as **Text**, and when the end-of-segment marker is CRLF, and when you use offset values to identify matching criteria beyond the first segment, you must be sure that the offset for the matching criteria for subsequent interchanges allows for the change in segment terminators from two (CRLF) to one (NL) character.

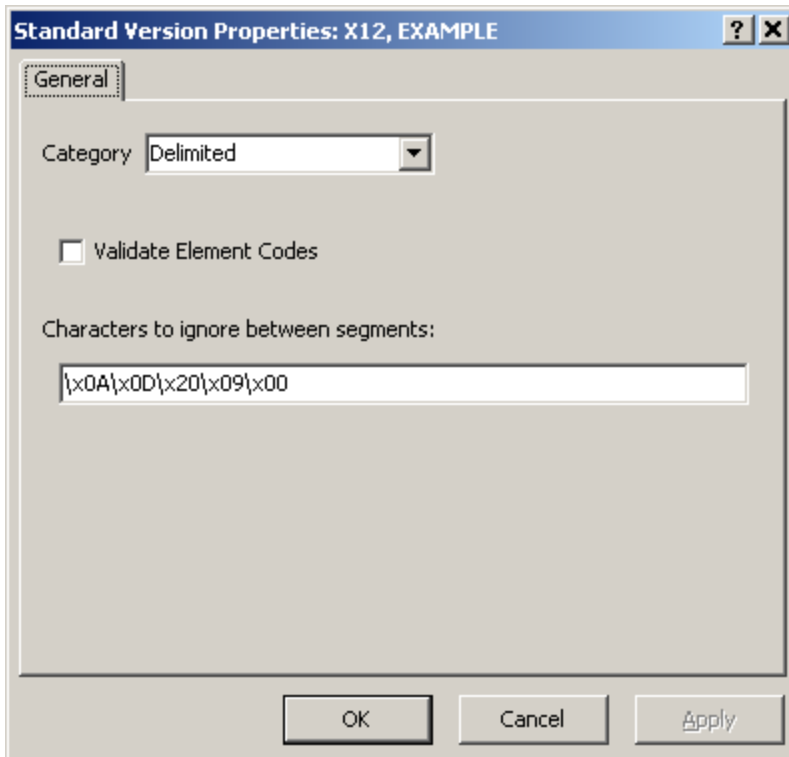
When the TRM parses the wrapper data, it places the information in the data element store fields for access during mapping, and in internal field locations for comparison during parsing. Some of the internal field information for the inbound wrapper is also accessible during mapping.

If the wrapper definition is a null wrapper, because there is no inbound wrapper to parse, there will be no wrapper information stored in the data element store or in the internal fields.

IMPORTANT: The result of comparisons and matching for such things as trade agreements and partners depends on what wrapper information is associated with which internal fields. In the wrapper definitions, certain wrapper elements have been linked to internal fields (Segment window). For example, if you have an incoming X12 ISA wrapper, the value in the ISA.06 (Interchange Sender ID) is linked to the internal field, Send Partner ID. To match inbound data, the elements in the segment must be associated with the appropriate internal field, otherwise no data is stored with which to match.

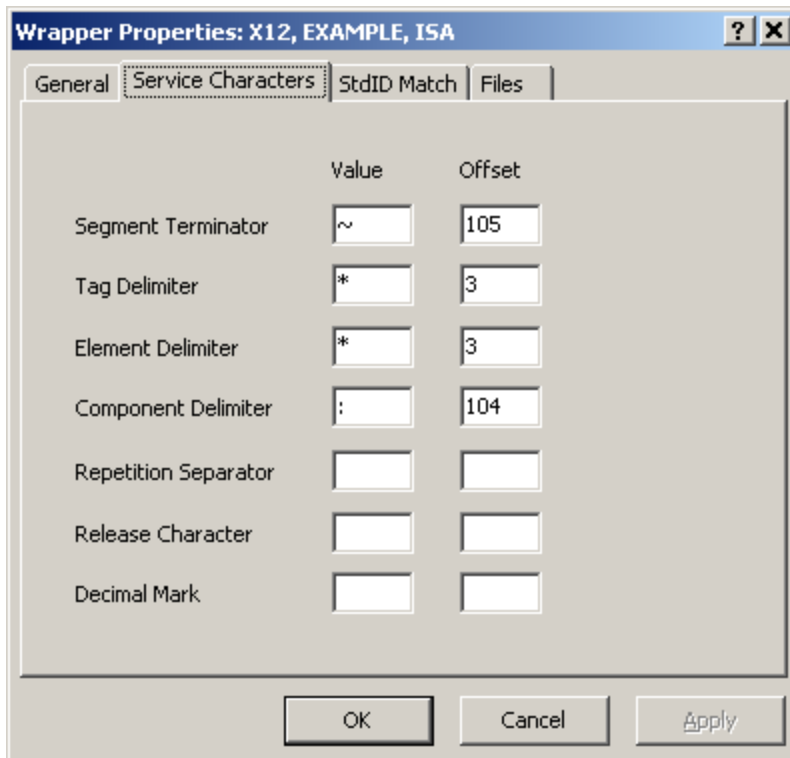
Parse and validate the incoming wrapper as follows:

- 1 Determine if the standard for this wrapper is delimited, fixed format, SWIFT, or XML by looking at the value in the Category field (Standard Version Properties window).



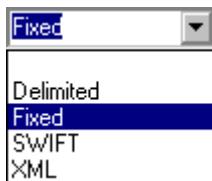
- If the standard version is delimited:

- Use the parsing routine for delimited data
 - and –
- Use the delimiters from the offset location
 - or –
- If no offset location is provided, use the delimiters given.




– or –

- If the standard is fixed, SWIFT or XML, use the appropriate parsing routine.



- Find the wrapper definition identified during standard identification to parse the data in the wrapper(s).



Wrapper: X12,EXAMPLE,ISA

Description: X12 Version 003030 wrapper

Header Segments:

Lvl	Seq	Seg Tag	Description	Rqmt
1	10	ISA	Interchange Control Header	M
2	20	GS	Functional Group Header	M

Contents:

Header: [Transaction Set Header](#)

Trailer: [Transaction Set Trailer](#)

Trailer Segments:

Lvl	Seq	Seg Tag	Description	Rqmt
2	30	GE	Functional Group Trailer	M
1	40	IEA	Interchange Control Trailer	M

- Parse and validate the input wrapper segments.

The remaining wrapper segments will be parsed and the information validated when the TRM encounters them as it sequentially processes the interchange, until it reaches the first document header.

NOTE: If your wrapper is defined as a null wrapper, because the inbound data has no wrapper, the TRM will continue to step 8 to look for acknowledgment profiles, and then to step 9 to look for a trade agreement. This is because there will be no partner definitions that have been stored for it to find.

Step 4. Look for Defined Partner IDs for the Sender and Recipient at this Level

Though the TRM will work without partner profiles, it checks for valid partner IDs at this point as follows:

- While parsing the interchange wrapper header, the TRM initializes internal partner fields (**Sender Partner Id**, **Sender Partner Qual**, **Sender Partner Int ID**, **Sender Int ID2**, **Rec Partner Id**, **Rec Partner Qual**, **Rec Partner Int ID**, and **Rec Partner Int ID2**) with inbound element values for any element with one of those internal fields defined. (The associated definition appears in the **Field** column on the Segment window).

NOTE: If partner fields are not defined on the interchange wrapper or if the corresponding element is omitted in the inbound data, then the internal field will be initialized to a null string.

Segment: X12, EXAMPLE, ISA

Description: Interchange Control Header

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	I01	Authorization Information Qualifier		M	ID	2	2
2	I02	Authorization Information		M	AN	10	10
3	I03	Security Information Qualifier	Password Qual	M	ID	2	2
4	I04	Security Information	Password	M	AN	10	10
5	I05	Interchange ID Qualifier	Send Partner Qual	M	ID	2	2
6	I06	Interchange Sender ID	Send Partner ID	M	AN	15	15
7	I05	Interchange ID Qualifier	Rec Partner Qual	M	ID	2	2
8	I07	Interchange Receiver ID	Rec Partner ID	M	AN	15	15
9	I08	Interchange Date	Date	M	DT	6	6
10	I09	Interchange Time	Time	M	TM	4	4
11	I10	Interchange Control Standards Identifier		M	ID	1	1
12	I11	Interchange Control Version Number	Interchange Versior	M	ID	5	5
13	I12	Interchange Control Number	Control Reference	M	NO	9	9
14	I13	Acknowledgment Requested	Acknowledgement F	M	ID	1	1
15	I14	Test Indicator	Test Indicator	M	ID	1	1

- 2 After parsing the interchange wrapper, the TRM will use the values for these internal fields to look for the sender and recipient partner IDs.
 - It first uses all 4 levels of IDs to find a unique partner.
 - otherwise –
 - If that match fails, it uses ID and Qual values allowing duplicates for a partner match.

Partner Explorer

Locations/StdID	Name	Interchange ID
Partners	Example Output Recipient Partner	TESTREC
Partner Relationships	Example Output Sending Partner	TESTSEND
Groups	Example X12 Recipient	ICH-REC-ID
	Example X12 Sender	ICH-SEND-ID

- 3 If it finds matching partners or the partner definitions are not required, MW Translator continues to the next step.
 - otherwise –

It proceeds as follows:

- a) When it cannot find the sender IDs, and if the **Partner Definition Required Sending Partner** box is checked (Standard ID window, Partners page), it generates an exception (reject).
- b) When it cannot find the recipient IDs, and if the **Partner Definition Required Recipient Partner** box is checked (Standard ID window, Partners page), it generates an exception (reject).

The screenshot shows a dialog box titled "Standard ID: X850TEST, 1, X12, EXAMPLE, ISA" with a "Partners" tab selected. The dialog has several sections:

- Location:** X850TEST
- Standard:** X12
- Version:** EXAMPLE
- Wrapper ID:** ISA
- Partner Definition Required:** Two checkboxes are checked: "Sending Partner" and "Recipient Partner".
- Default Sending Partner:** A dropdown menu labeled "Name:" is currently empty.
- Default Recipient Partner:** A dropdown menu labeled "Name:" is currently empty.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Step 5. Find Partner Profiles for the Sender and Recipient Based on Partner IDs

The TRM is not required to use partner profile definitions, unless the **Partner Definition Required** boxes are checked (Standards ID window, Partners page) for either the sender or recipient. However, it attempts to find the partner IDs first to determine if it has explicit configurations it must use before it attempts another strategy.

The TRM proceeds as follows:

- 1 After parsing the interchange wrapper, MW Translator uses the values for these internal fields to look for the sender and recipient partner profiles.
 - a) If the TRM finds the sender profile, then it uses that profile.
 - b) If it finds the recipient profile, then it uses that profile.

- 2 If it does not find the sender or recipient profile and the associated **Partner Definition Required** box is not checked, it determines the profiles to use as follows:
- If both the **ID** and **Qual** are null (string with a length of zero) in the internal table, and the corresponding default partner is defined for the Standard ID, then it uses that default partner.
 - otherwise –
 - It does not use a partner profile.

Standard ID: X850TEST, 1, X12, EXAMPLE, ISA

General | **Partners** | Routing | Trade Agreements | Acknowledgments

Location: X850TEST
 Standard: X12 Version: EXAMPLE Wrapper ID: ISA

Partner Definition Required
 Sending Partner Recipient Partner

Default Sending Partner
 Name:

Default Recipient Partner
 Name:

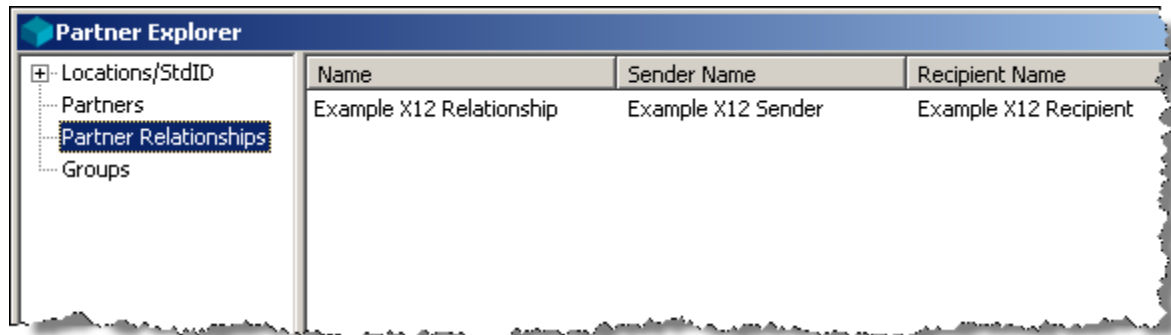
OK Cancel Apply

The processing report indicates the partner information that matched the incoming data.

```
>>> Wrapper set definition file name:
C:\MWTranslator\DefaultEnv\Cfg\x12\W-X12-EXAMPLE-ISA.txt <<<
Processing wrapper, level: 1    Control Reference: 000010001
Sender ID: ICH-SEND-ID    Qual: ZZ    Key: 1509681622
Recipient ID: ICH-REC-ID    Qual: ZZ    Key: 1216895274
```

Step 6. Find a Partner Relationship Definition, If One Exists

If partner profiles exist for both the sender and the recipient, the TRM looks for a specific relationship between these partners to use instead.



The screenshot shows the 'Partner Explorer' window. On the left is a tree view with 'Locations/StdID', 'Partners', 'Partner Relationships' (highlighted), and 'Groups'. On the right is a table with the following data:

Name	Sender Name	Recipient Name
Example X12 Relationship	Example X12 Sender	Example X12 Recipient

Step 7. Repeat Steps 3-6 for Remaining Wrapper Levels 2 through 4, if They Exist

Repeat the previous 4 steps for any remaining levels of wrappers in order to match the incoming data with partner profiles and partner relationships.

The processing report indicates the partner information that matched the incoming data. If a relationship is found, it specifies the relationship.

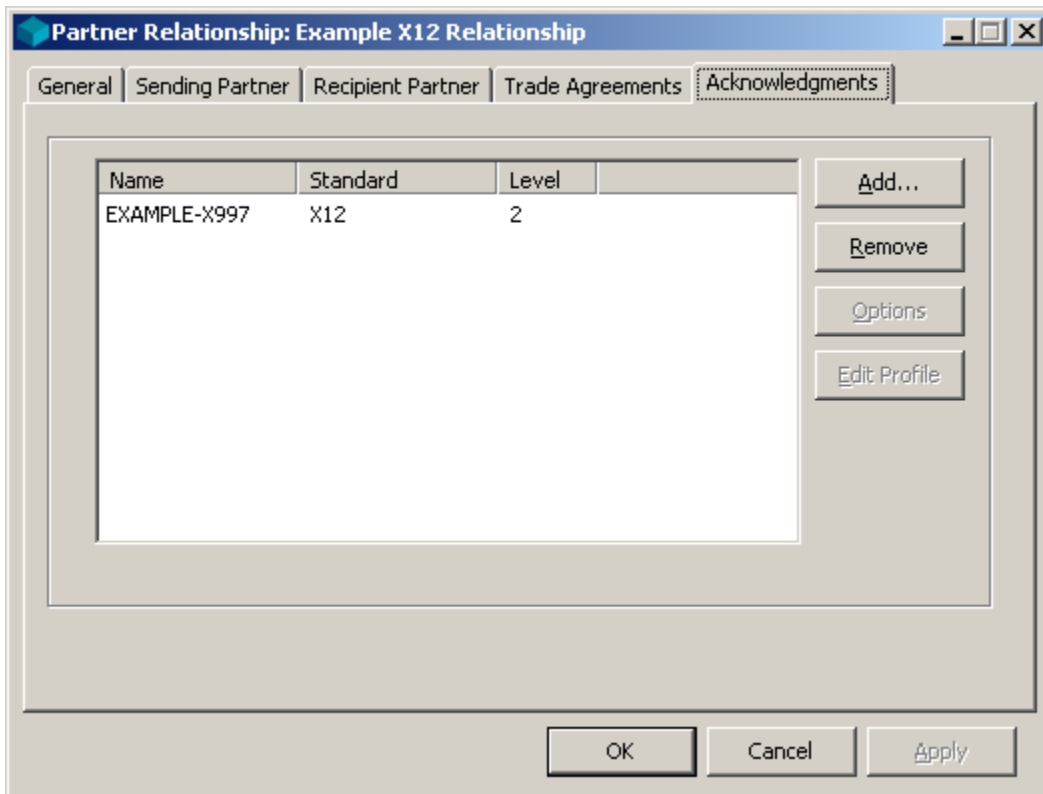
```
Processing wrapper, level: 2   Control Reference: 100010001
Partner Relationship Name: Example X12 Relationship
Sender Name: Example X12 Sender
      ID: FG-SEND-ID   Key: 2060521
Recipient Name: Example X12 Recipient
      ID: FG-REC-ID   Key: 6303532
```

Step 8. Find the Acknowledgment Profiles, if They Exist

To create an acknowledgment, the TRM searches for a maximum of 9 acknowledgment profiles identified on the **Acknowledgment** tab, each of whose input wrapper matches that of the inbound interchange. This list might be associated with one of 3 different configurations.

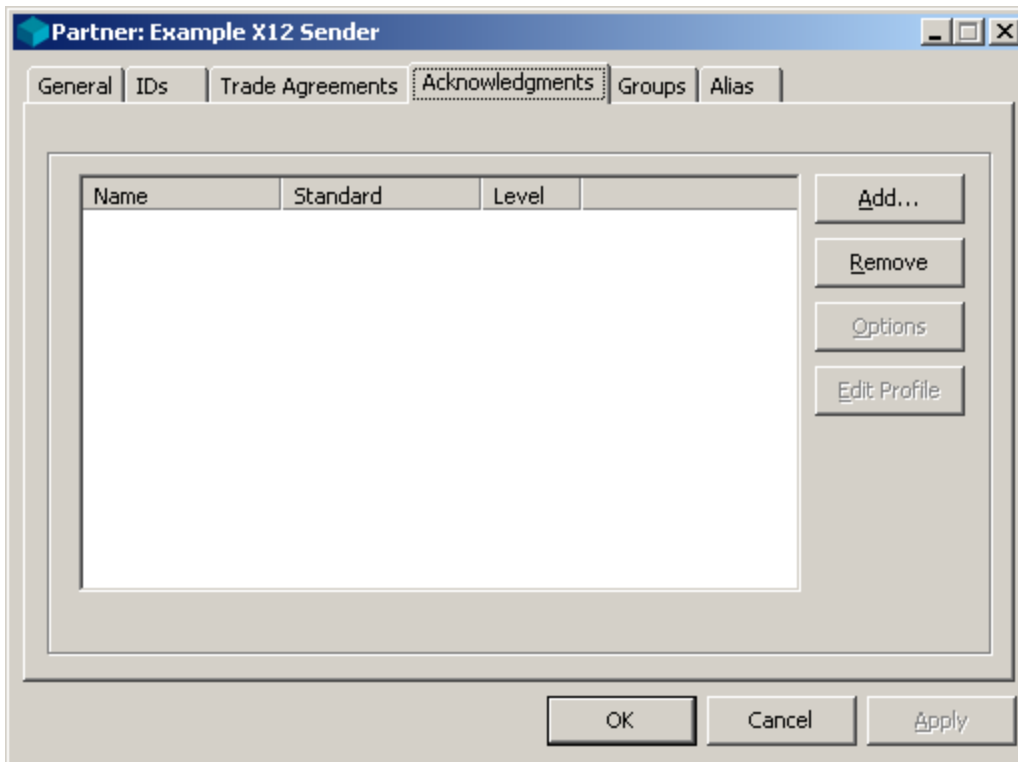
- 1 Search for acknowledgments assigned to a partner relationship (**Partner Relationships** folder).
Generate any acknowledgments on the list that match the wrapper definition used during standard identification.
– otherwise –

If there is nothing on this list, the TRM will not generate acknowledgments.



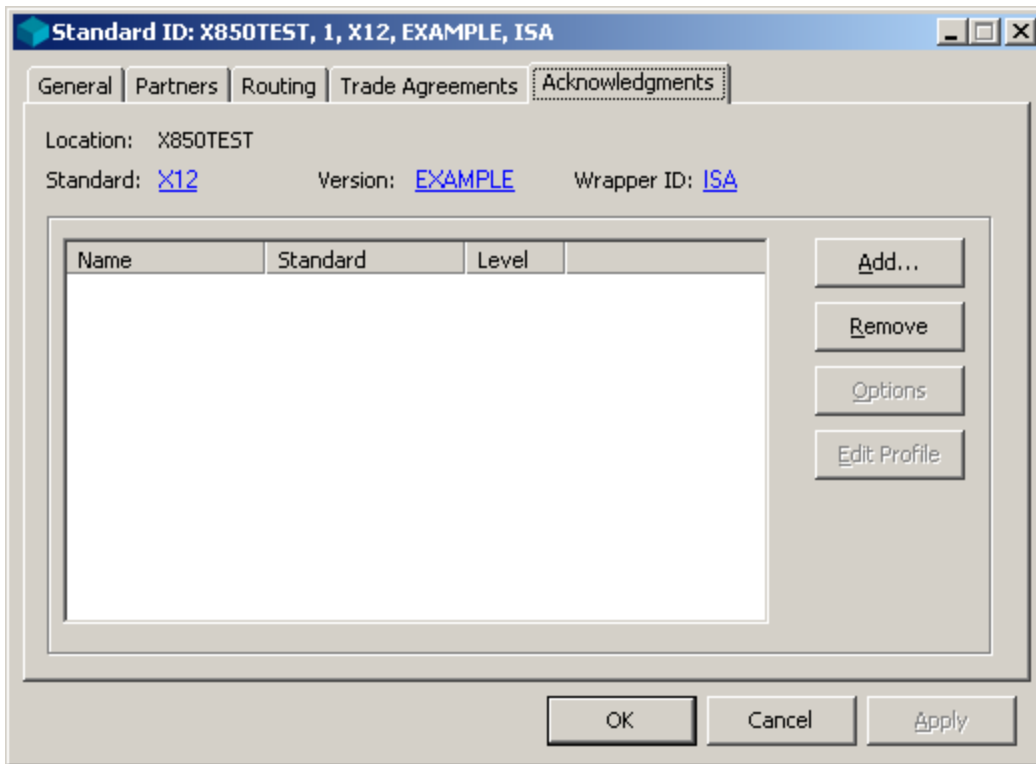
- 2 If there is no partner relationship, search for acknowledgments assigned to the individual partner profile of the sender (**Partners** folder). Generate up to 9 acknowledgments on the list.
– otherwise –

If there is nothing on this list, the TRM will not generate acknowledgments.



- 3 If there is no partner profile for the sending partner, search for acknowledgments assigned to the Standard ID.
– otherwise –

If there is nothing on this list, the TRM will not generate acknowledgments.



Step 9. Find a Trade Agreement Profile

This is the second of three tasks that the TRM must perform. It must find a trade agreement profile. The TRM makes an extraordinary effort to find a valid trade agreement in order to continue processing.

First it scans the document header, which is identified in the **CONTENTS** Header and Trailer boxes (Wrapper window). Then it looks for an appropriate Trade Agreement profile. Multiple trade agreements can be associated with a given partner. The TRM attempts to match a trade agreement profile whose value in the **Identified Standard** field matches the standard identified for the incoming wrapper during the standard identification process.

It then attempts to match values from the incoming wrapper assigned to specific internal fields against the values in certain fields of the Trade Agreement profile records. Note the following assignments for **Agency**, element 7, and **Document Version**, element 8, in the X12 EXAMPLE GS segment.

Segment: X12, EXAMPLE, GS

Description: Functional Group Header

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	479	Functional Identifier Code	Functional Group ID	M	ID	2	2
2	142	Application Sender's Code	Send Partner ID	M	AN	2	15
3	124	Application Receiver's Code	Rec Partner ID	M	AN	2	15
4	373	Date	Date	M	DT	6	6
5	337	Time	Time	M	TM	4	8
6	28	Group Control Number	Control Reference	M	NO	1	9
7	455	Responsible Agency Code	Agency	M	ID	1	2
8	480	Version / Release / Industry Identifier Code	Document Version	M	AN	1	12

Note also the assignment for the **Document ID**, element 1, in the ST segment.

Segment: X12, EXAMPLE, ST

Description: Transaction Set Header

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	143	Transaction Set Identifier Code	Document ID	M	ID	3	3
2	329	Transaction Set Control Number	Control Reference	M	CR	4	9

During parsing, the TRM places the incoming values for elements in their assigned internal fields. It then uses any values stored in these fields to find a trade agreement by matching them with the values defined for various trade agreements. Note the values on the **Matching Fields** tab on the Trade Agreement Profile window.

The screenshot shows a window titled "Trade Agreement Profile: EXAMPLE-X850" with a "Matching Fields" tab selected. The "Identified Standard" is set to "X12". Under "Input Field Values", the following fields are populated: "Doc Id" is 850, "Version" is 003030, "Agency" is X, and "Assoc" is empty. The "Release" field is also empty. The "OK", "Cancel", and "Apply" buttons are visible at the bottom.

This matching schema is given in the following table:

Matching Values (Location of Definition)	Trade Agreement Profile, Matching Fields Page
Standard for identified wrapper (Standard ID window)	Identified Standard
Document ID (Segment window, Field)	Doc Id
Document Version (Segment window, Field)	Version
Release (Segment window, Field)	Release
Agency (Segment window, Field)	Agency
Association Code (Segment window, Field)	Assoc

Matching always occurs between values in the matching fields of the trade agreement profile and the internal fields to which elements are assigned, which are visible on the Segment window.

NOTE: If there is no value in a particular **Matching Field** on the Trade Agreement profile, the value in the internal field must be null to match. The value in the internal field will be null either when no data exists for the incoming element or when the element is not assigned to the internal field.

The following decision matrix shows when a match occurs:

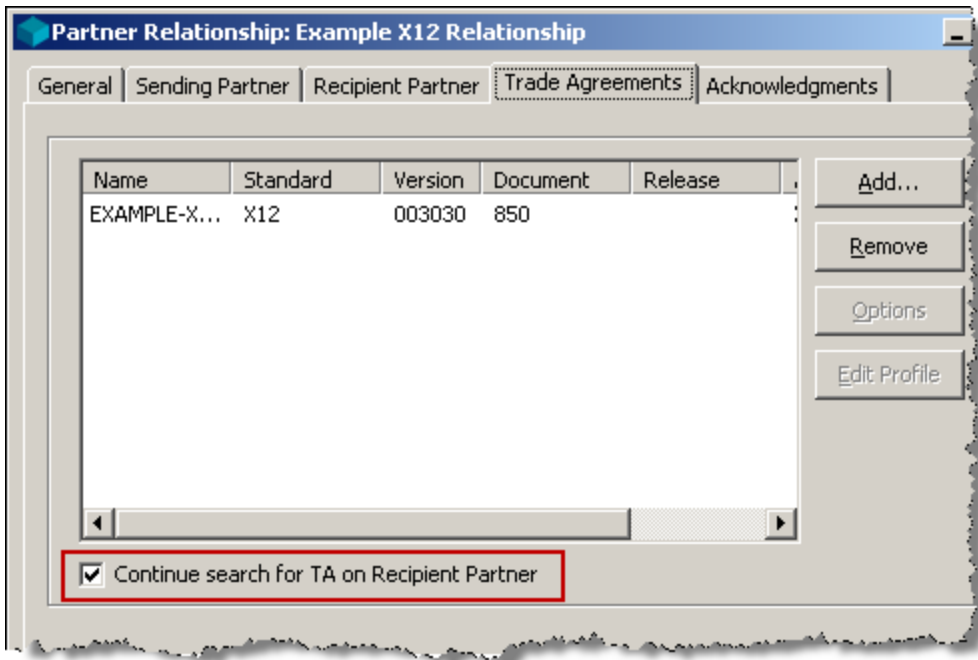
Input Data Exists	Element Assigned to Field (Segment)	Value in Matching Field (TA Profile)	Match Occurs (if values match)
No	No	No	Yes
Yes	No	No	Yes
No	Yes	No	Yes
Yes	Yes	Yes	Yes
No	No	Yes	No
Yes	Yes	No	No
No	Yes	Yes	No
Yes	No	Yes	No

Using this matching strategy, the TRM proceeds as follows to find a trade agreement. Note that it will begin with step 1, 2, or 3 depending on the partner definitions it has found, if any. It will execute step 1 only if it has already found a partner relationship. It will execute step 2 only if it has found a valid partner profile for the recipient. If it has found no valid partner, it only executes step 3.

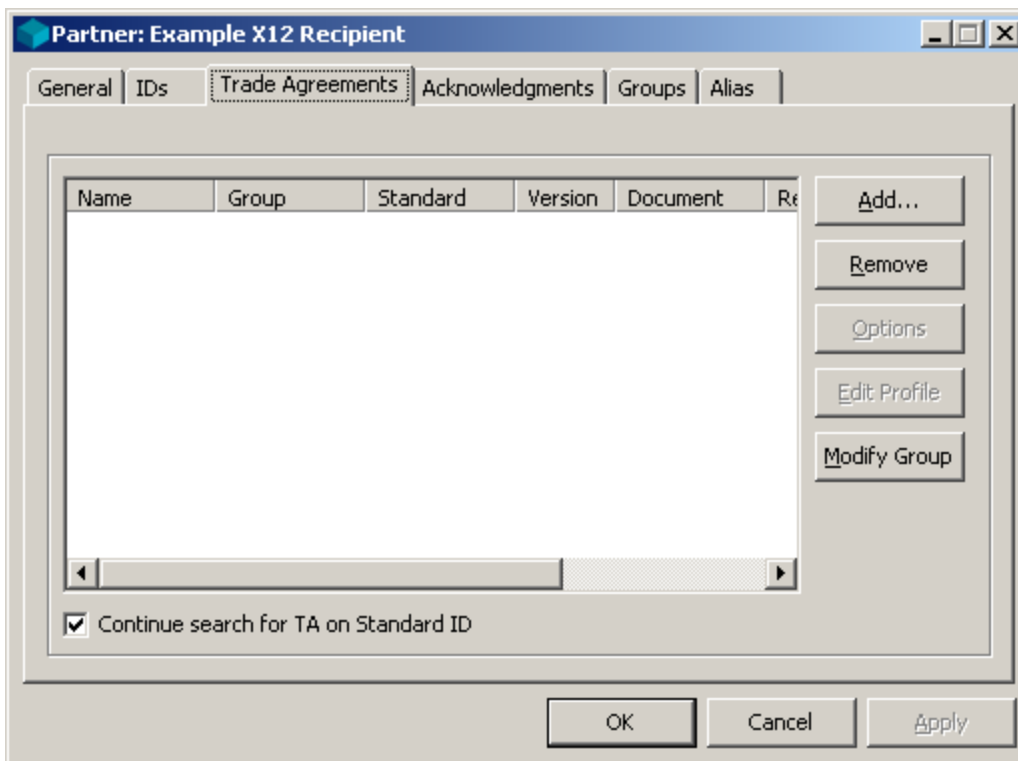
IMPORTANT: The TRM searches the list for a matching trade agreement in a random order. The first trade agreement on the list may not be the first one that it checks.

- 1 Search for trade agreements assigned to a partner relationship (Partner Relationships folder). Use the trade agreement on the list that matches first, using the criteria given above.
 - If the TRM does not find a matching trade agreement and **Continue search for TA on Recipient Partner** is checked, it goes to the next option, to search for trade agreements assigned to the recipient partner.
 - otherwise –

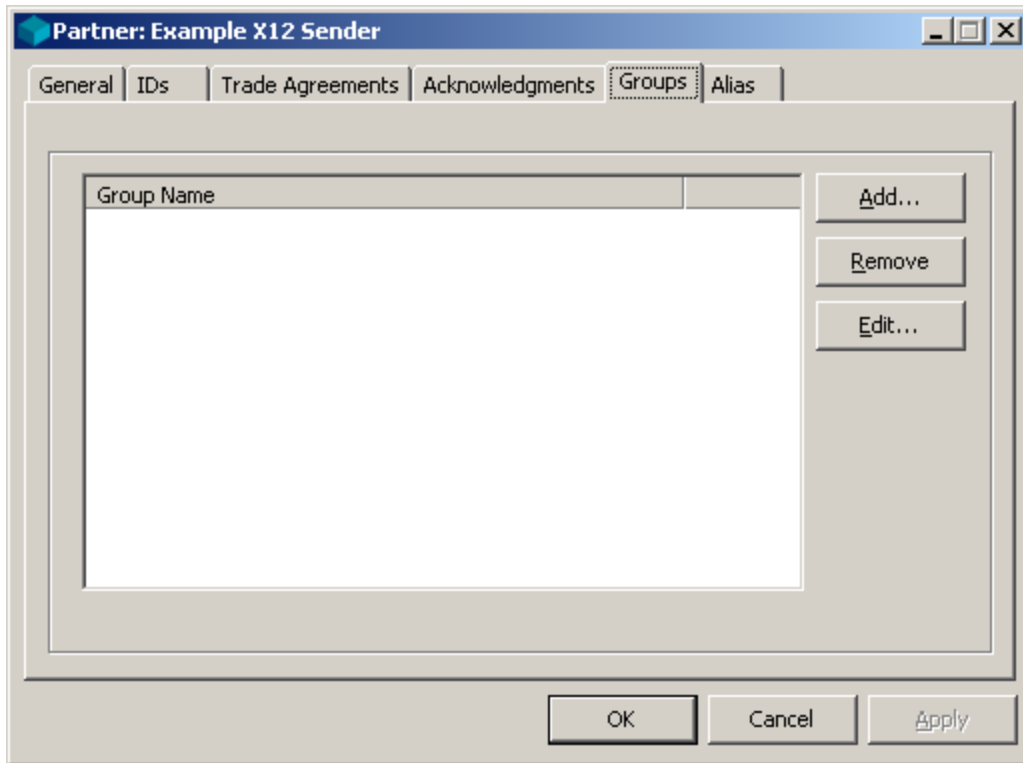
- If **Continue search for TA on Recipient Partner** is not checked, it stops processing.



- 2 Search for trade agreements assigned to the individual partner profile for the recipient (**Partners** folder). Use the trade agreement on the list not associated with a group that matches first, using the criteria given above.

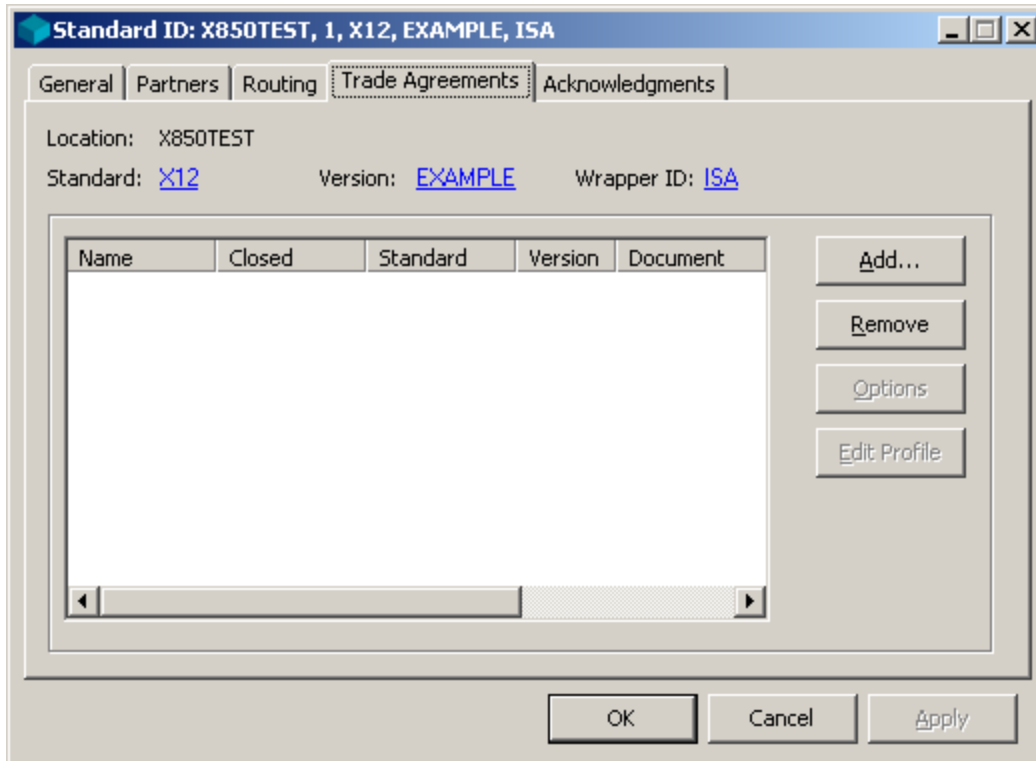


- If a matching trade agreement is not found on the list, search the list for a trade agreement associated with a group on the sending partner's list of groups. Repeat for all groups on the sending partner's list.



- If the TRM does not find a matching trade agreement, and if **Continue search for TA on Standard ID** is checked, it goes to the next option to search for trade agreements associated with the standard ID.
 - otherwise –
 - If **Continue search for TA on Standard ID** is not checked, it stops processing.
- 3** Search for trade agreements assigned to the Standard ID, **Trade Agreements** tab.

- a) Use the trade agreement on the list not associated with a group that first matches using the criteria given above.



- b) If a matching trade agreement is not found on the list, search the list for a trade agreement associated with a group on the sending partner's list of groups. Repeat for all groups on the sending partner's list.
- 4 If the TRM does not find a matching trade agreement here, it repeats steps 1-3 with the internal fields **Document Id, Document Version, Release, Agency and Association Code** set to blank, thus overriding any values originally stored from the input document. To match, a default trade agreement for the identified standard must exist where all values in the **Input Field Values** boxes would be blank.

NOTE: In this case, the inbound document is not known and any matching trade agreement should have an action to validate wrappers only. Since it would not know what document it might have, any attempt to validate contents or translate would most likely abort.

- 5 If the TRM is unable to find a matching trade agreement profile, it stops processing.

– otherwise –

If the TRM finds a matching trade agreement profile, it will know what action it must perform: route file, route file without header, validate wrapper, validate contents or translate.

- If it only has to route the file or validate the wrapper, which it has already done in order to reach this point, it will skip the next steps and attempt to determine where to send the document.
- otherwise –

- If it must validate the contents or translate the document, the TRM proceeds to the next step.

The processing report lists the trade agreement it found and the definition with which this trade agreement is associated. The report also shows information that it parsed from the inbound header wrapper: in this case, the document ID, the version, the agency and the control reference.

```
>>> Document definition file name:  
C:\MWTranslator\DefaultEnv\Cfg\x12/D-X12-EXAMPLE-850.txt <<<  
Trade Agreement found on Partner Relationship: EXAMPLE-X850  
Identified document: 850 Version: 003030 Agency: X  
Control Reference: 0001
```

Step 10. Parse the Incoming Document(s) (Contents Validation and Translation)

In order to parse the incoming document, the TRM should have identified the incoming document (seen in the Trade Agreement Profile window, **Input** tab). Otherwise, it typically would not know what document definition to use for parsing.

The TRM processes data one segment at a time as it looks for one of the following:

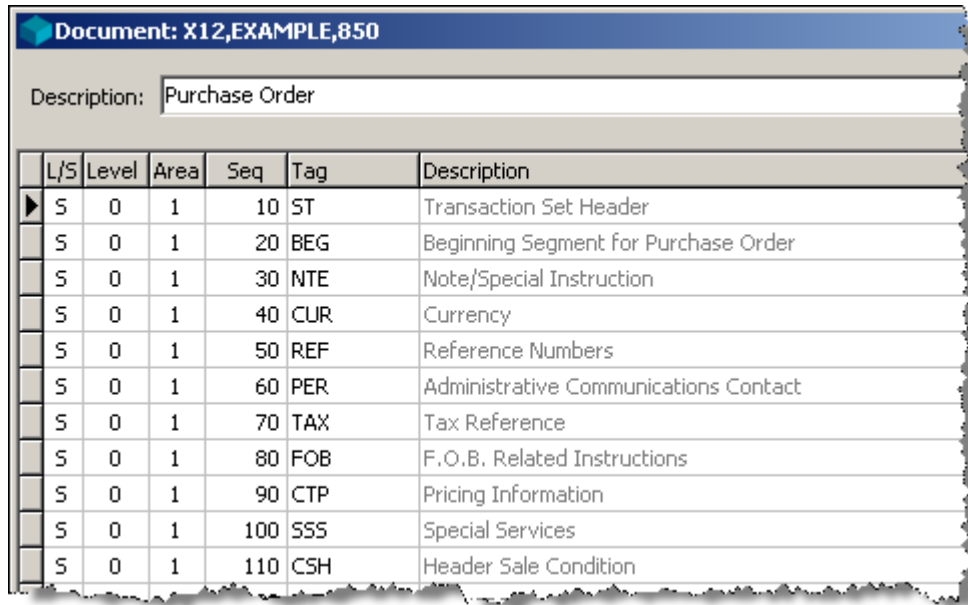
- A document trailer segment
- Another valid wrapper segment
- Another instance of the document header segment

If you are validating wrapper segments only, refer to the topic, *Step 3. Parse and Validate One Level of the Incoming Wrapper* (on page 50).

The screenshot shows a dialog box titled "Trade Agreement Profile: EXAMPLE-X850" with four tabs: "Matching Fields", "Input", "Options", and "Outputs". The "Input" tab is selected. Inside the dialog, there is a section for "Inbound Document" with a small icon and a table of fields. The table has three rows: "Standard: X12", "Version: EXAMPLE", and "Doc Id: 850". The "Doc Id" field is highlighted with a red border. To the right of the "Inbound Document" section are two text input fields: "Security Document:" and "Security User Exit:". At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Inbound Document	
Standard:	X12
Version:	EXAMPLE
Doc Id:	850

The TRM uses the standards definitions for the incoming document based on the standard identified for the wrapper and the incoming document ID. When the TRM parses the contents, the values are placed in the data element store for use during mapping.



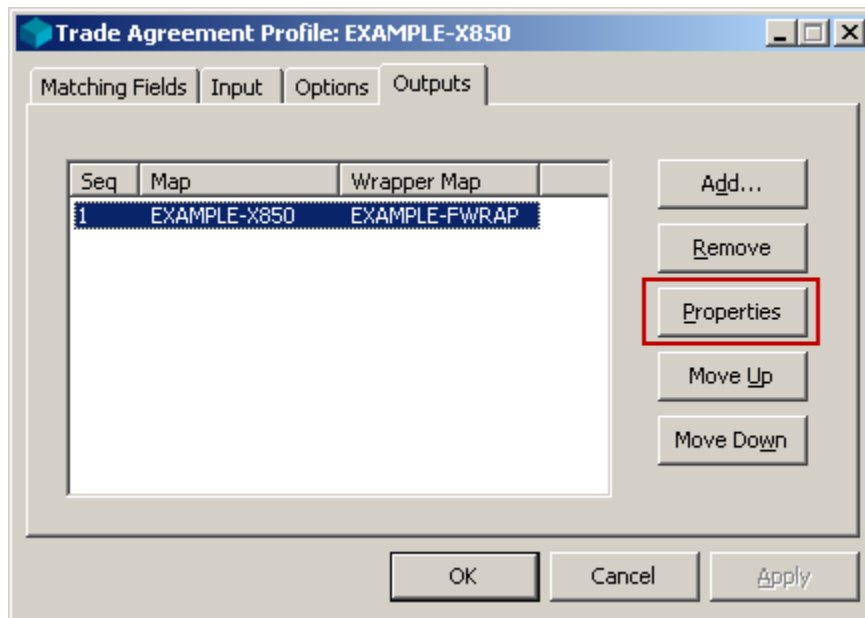
	L/S	Level	Area	Seq	Tag	Description
▶	S	0	1	10	ST	Transaction Set Header
	S	0	1	20	BEG	Beginning Segment for Purchase Order
	S	0	1	30	NTE	Note/Special Instruction
	S	0	1	40	CUR	Currency
	S	0	1	50	REF	Reference Numbers
	S	0	1	60	PER	Administrative Communications Contact
	S	0	1	70	TAX	Tax Reference
	S	0	1	80	FOB	F.O.B. Related Instructions
	S	0	1	90	CTP	Pricing Information
	S	0	1	100	SSS	Special Services
	S	0	1	110	CSH	Header Sale Condition

If you are only validating the wrappers and not the contents or translating the document, you may not have identified a document ID.

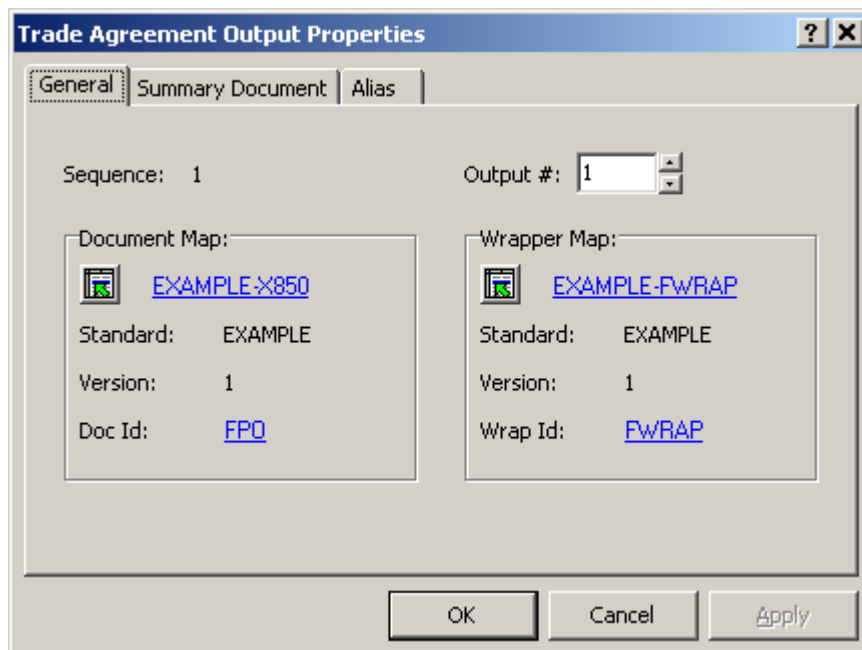
Step 11. Generate Outbound Documents



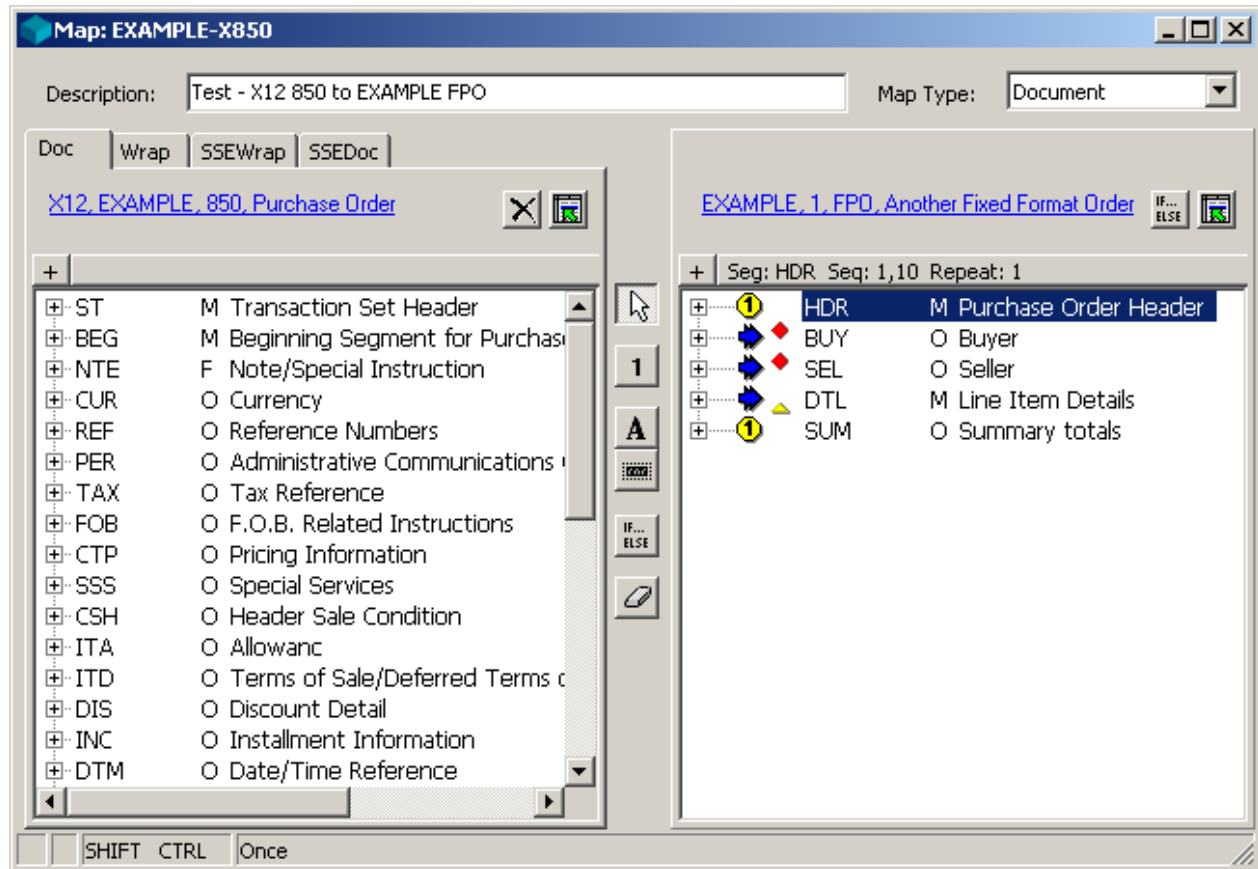
From the Trade Agreement Profile window, **Outputs** tab, select the **Properties** button to view the properties.

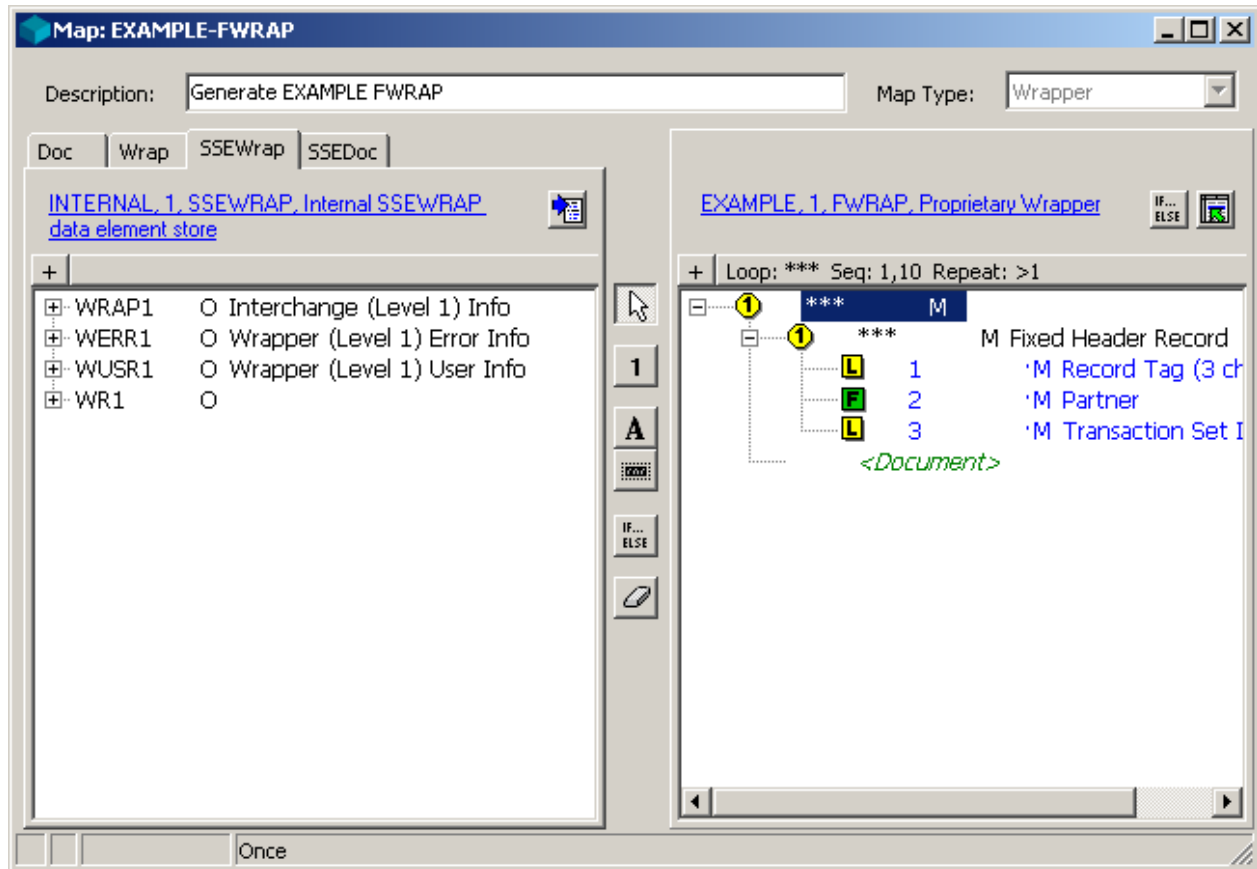


The TRM uses the maps defined on the Trade Agreement Output Properties window, **General** tab, to create the output documents.



NOTE: The instructions in the wrapper map, if one exists, are executed as they are needed. The wrapper instructions that pertain to the entire wrapper and to header segments are executed before any instructions in the document map. The wrapper instructions that pertain to trailer segments are executed after instructions in the document map.





The processing report indicates the output file it created as the first of many possible output files that might result from processing an input file. If you are showing the file names for this environment (Modify Options window, **General** tab), you will see the names of the definition text files used for the following:

- Outbound wrapper
- Wrapper map
- Outbound document
- Document map

The translation output stream is the name of the file for the specified output number. You can also see the source and destination locations. Following this information is the name of the control reference file that it uses to generate control numbers. It also shows the name of the map in the configuration database that it used to create the output document. After the processing information, it shows the status of the input document.

```
>>> Wrapper set definition file name:
C:\MWTranslator\DefaultEnv\Cfg\appl\W-EXAMPLE-1-FWRAP.txt <<<
>>> Map file name: C:\MWTranslator\DefaultEnv\Cfg\map\M-EXAMPLE-FWRAP.txt <<<
>>> Document definition file name:
C:\MWTranslator\DefaultEnv\Cfg\appl\D-EXAMPLE-1-FPO.txt <<<
>>> Map file name: C:\MWTranslator\DefaultEnv\Cfg\map\M-EXAMPLE-X850.txt <<<
Translation Output stream created: C:\MWTranslator\DefaultEnv\Out\BU08MUO.txt
Output number: 1 Source: TESTSEND-MAILBOX Destination: TESTREC-MAILBOX

>>> Control reference file name: C:\MWTranslator\DefaultEnv\Cfg\CTRLREF.TXT <<<
Map Name: EXAMPLE-X850

Document Status: A Control Reference: 0001
```

Step 12. Repeat Steps 9-11 for Remaining Documents

As the TRM encounters more document headers, it will repeat steps 9-11 until it has processed all the documents.

The TRM must also determine when to break a stream to create new outbound files that can be properly routed. For more information about breaking, refer to the topic, *Step 17. Repeat steps 14 and 16 for each defined output (translation)* (on page 97).

The processing report indicates the additional trade agreements it uses, based on the documents it found. If the trade agreement remains the same, the report will not repeat the names of the files it uses. After all documents have been processed, it shows the status of the wrapper that contains the documents.

```
Trade Agreement found on Partner Relationship: EXAMPLE-X850
Identified document: 850 Version: 003030 Agency: X
Control Reference: 0002

Map Name: EXAMPLE-X850

Document Status: A Control Reference: 0002

Trade Agreement found on Partner Relationship: EXAMPLE-X850
Identified document: 850 Version: 003030 Agency: X
Control Reference: 0003

Map Name: EXAMPLE-X850

Document Status: A Control Reference: 0003

Wrapper level 2 Status: A Control Reference: 100010001
```

Step 13. Determine IDs for Outbound Partners

The TRM uses partner profiles to generate the sender and recipient IDs for the outbound wrappers. By default, the outbound partner records used will be the same as the inbound partners.

However, you may get unexpected results when input values do not match wrapper definitions and when partner records do not have to match, as explained in *Step 4* (on page 53). Under these circumstances, MW Translator goes to great lengths to populate the internal fields for the outbound partner IDs as follows:

- 1 MW Translator first tries to match the inbound partner values with partner definitions for the outbound level 1 (interchange) wrapper.

Example: Interchange receiver ID (ABC) and qualifier (ZZ) do not match any partner defined with these values at the interchange level.

- 2** If no match is made, it then tries to match the inbound partner values with outbound partner definitions for the level two (functional group).
Example: Functional group receiver ID (ABC) does not match any partner defined with the same values at the functional group level.
- 3** If no match is made at the functional group level, MW Translator uses the value from the functional group level wrapper to try to match a partner defined at the parent level, which in this case is the interchange level.
Example: Functional group receiver ID (ABC) does match a partner defined with the same values at the parent, interchange level.
- 4** If a match is made, MW Translator then populates both the level one and level two outbound wrapper fields with the value in the matched partner definition.
Example: Result is that the values from the interchange level of partner ABC are used to populate the interchange and functional group fields of the outbound wrapper. The inbound interchange-level values of ABC/ZZ were never stored in the outbound internal fields, because they did not match values defined at the interchange level for any partner record.

You can override the default behavior and provide alternative IDs in one of two ways:

- Specify partner IDs on the Trade Agreement Options window for a trade agreement associated with a partner relationship, partner, or standard ID
- or –

- Use an alias type on the Trade Agreement Profile that matches an alias partner for a particular partner

The TRM looks for alternative sender and recipient IDs as follows:

- 1 Using the trade agreement profile it found earlier, the TRM searches the Trade Agreement Options for alternative sending partner and recipient partner IDs for the outbound wrapper. If there is any information missing, and if this is not a partner relationship, it will proceed to the next option, to use aliases.

Partner Relationship Trade Agreement Options: EXAMPLE-X850 ? X

Output: 1,EXAMPLE,1,FPO 1 of 1

Sending Partner Recipient Partner Misc

Partner Name: Example Output Sending Partner

Location:

Override Location: TESTSEND-MAILBOX

Interchange

ID: TESTSEND Qual:

Int ID: Int ID2:

Functional Group

ID: Qual:

Int ID: Int ID2:

OK Cancel Apply

The screenshot shows a dialog box titled "Partner Relationship Trade Agreement Options: EXAMPLE-X850". It has a tabbed interface with three tabs: "Sending Partner", "Recipient Partner", and "Misc". The "Recipient Partner" tab is active. At the top, there is an "Output:" dropdown menu showing "1,EXAMPLE,1,FPO" and "1 of 1". Below the tabs, there are several input fields:

- Partner Name:** A dropdown menu with "Example Output Recipient Partner" selected. This field is highlighted with a red box.
- Location:** An empty text input field.
- Override Location:** A text input field containing "TESTREC-MAILBOX".
- Interchange:** A section containing:
 - ID:** A text input field with "TESTREC" entered. This field is highlighted with a red box.
 - Qual:** An empty text input field.
 - Int ID:** An empty text input field.
 - Int ID2:** An empty text input field.
- Functional Group:** A section containing:
 - ID:** An empty text input field.
 - Qual:** An empty text input field.
 - Int ID:** An empty text input field.
 - Int ID2:** An empty text input field.

At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Apply".

- 2 Search for any missing information using aliases as follows:
 - a) Determine a sender and recipient alias using the alias type defined in the Trade Agreement Output Properties window, **Alias** tab.

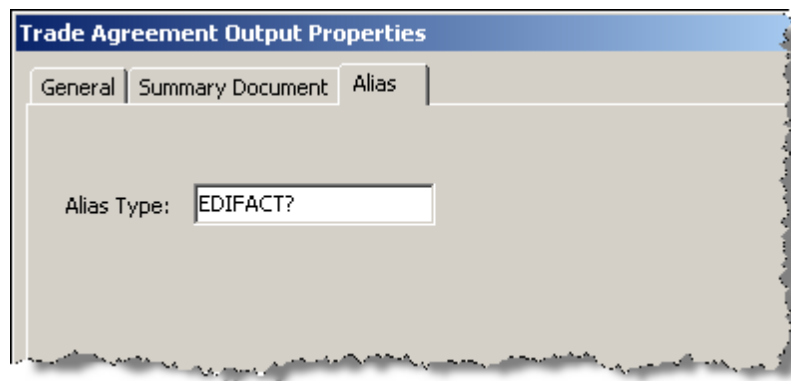
The screenshot shows a dialog box titled "Trade Agreement Output Properties". It has three tabs: "General", "Summary Document", and "Alias". The "Alias" tab is active. Below the tabs, there is a single input field labeled "Alias Type:" containing the text "APPL1".

- If the alias type contains the character, *?*, then replace the *?* with *S* to create the sender alias type and replace the *?* with *R* to determine the recipient alias type.

With two distinct alias types, you can use aliases to find both outbound sender and recipient partners when only one (sender or recipient) partner exists on input. Typically, this is used to translate proprietary formats that provide only the recipient ID, to map to valid X12 or EDIFACT sender and receiver IDs.

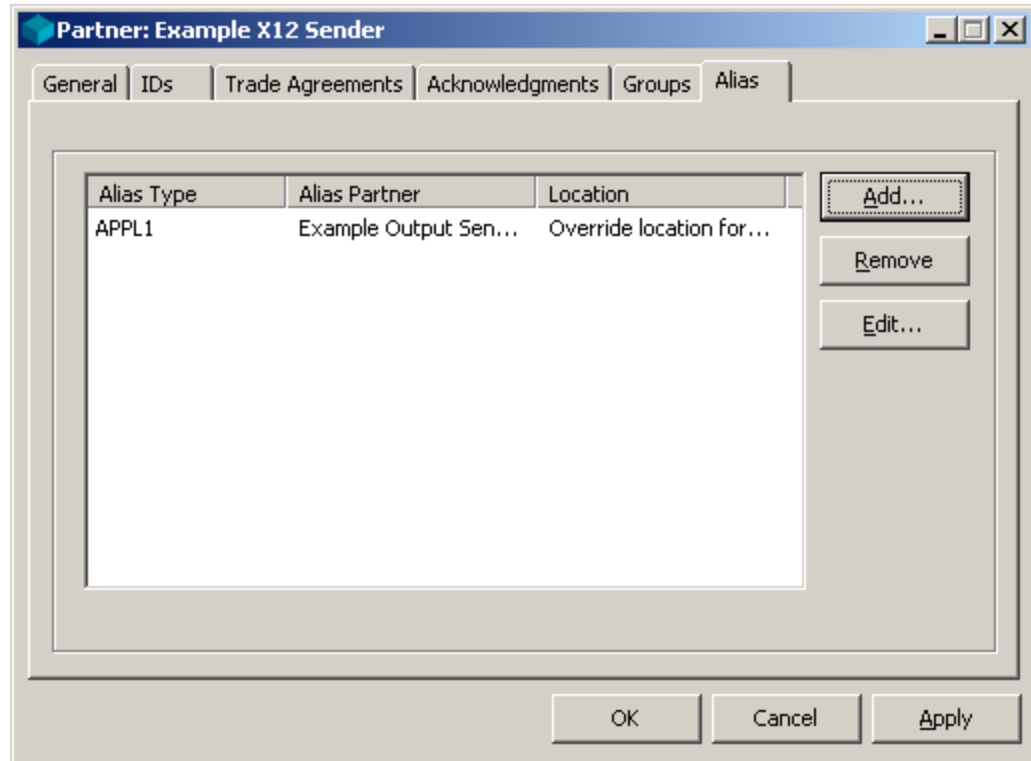
– otherwise –

- If the alias type does not contain the character, *?*, then the TRM uses the alias type unchanged for both the sender and recipient alias type.

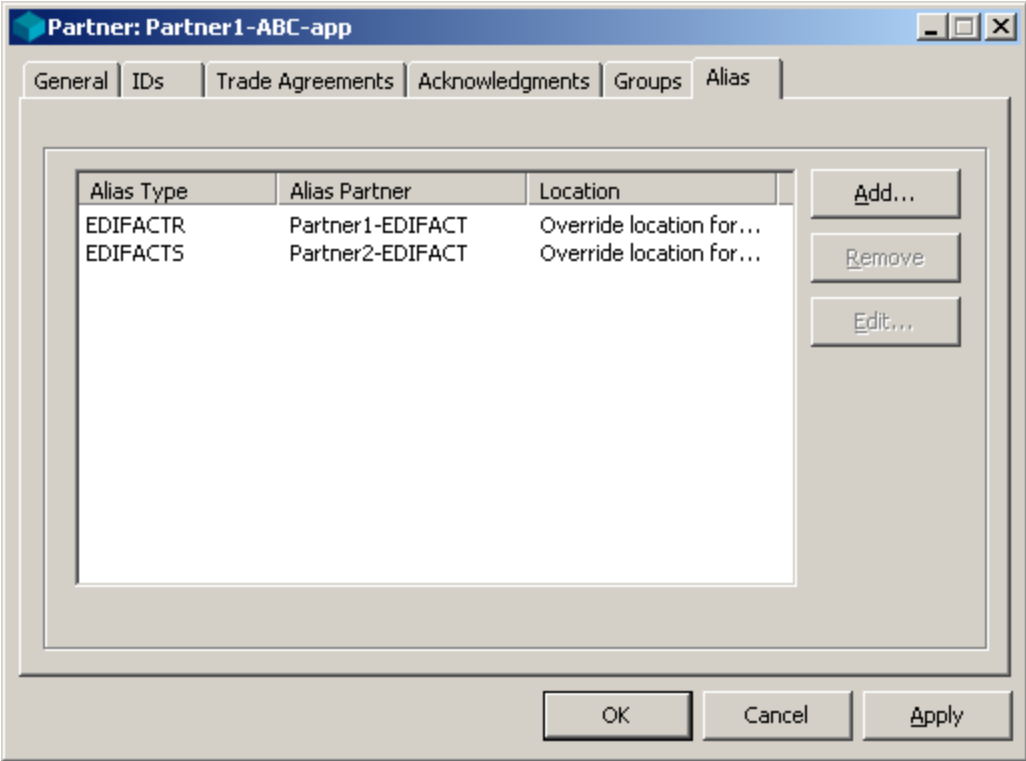


- b) Find the outbound sending partner.

- If an inbound sending partner exists, search for the sender alias type in the inbound sender alias list.

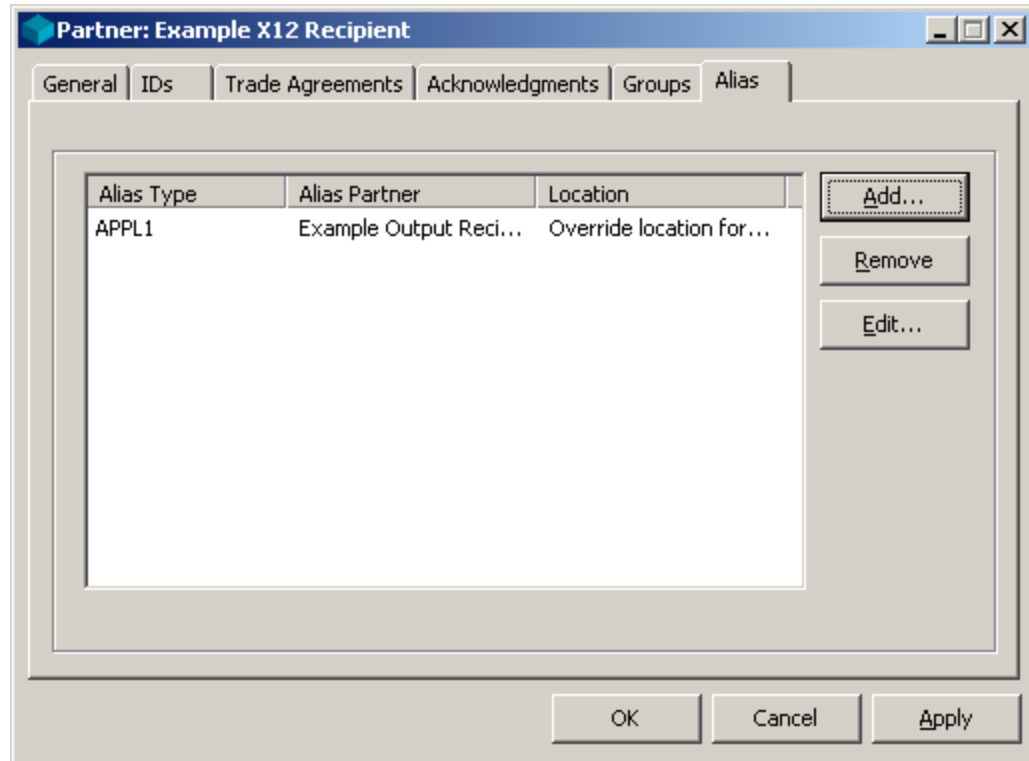


- If the sender does not exist, but the recipient does, and if the alias type of the trade agreement contains a ?, replace the ? with S and search for this sender alias type in the inbound recipient alias list. If no alias is found, proceed to the next option.

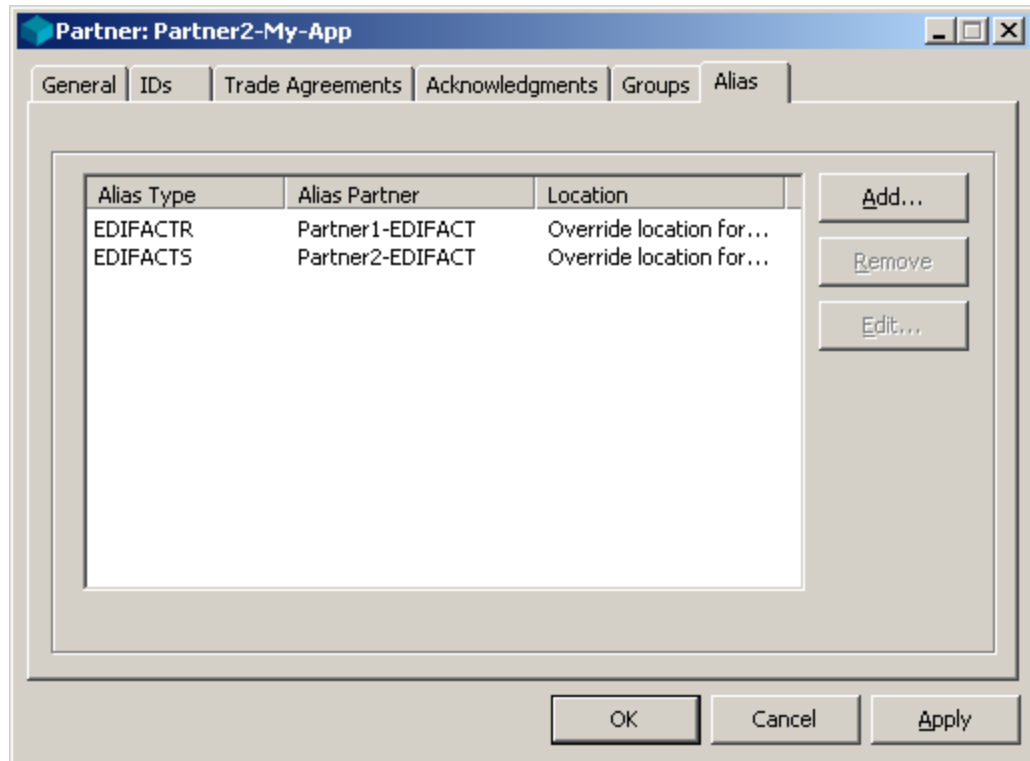


c) Find the outbound recipient partner.

- If an inbound recipient partner exists, search for the recipient alias type in the inbound recipient alias list.



- otherwise –
- If the recipient partner does not exist, but the sender does, and if the alias type of the trade agreement contains a ?, replace the ? with *R* and searches for this recipient alias type in the inbound sender alias list.
 - otherwise –
- If no alias is found, proceed to the next option.



NOTE: Aliasing does not affect the creation of control documents (acknowledgments). To create acknowledgment wrapper(s), the roles of sender and recipient values are reversed. The outbound recipient partner values received in the input wrapper(s) always become the inbound sending partner values in the acknowledgment wrapper(s). Similarly, the outbound sending partner values received in the input wrapper(s) always become the inbound recipient partner in the acknowledgment wrapper(s). For routing and validation, there is no such switching. The outbound partners maintain the same roles (sender and recipient) as the inbound partners. You only use aliasing when you need to substitute sender/receiver values in the translated output document(s) or specify alternative source and destination locations.

- 3 If there is still information missing, look for the missing definitions within the inbound definitions, which may mean a Partner Relationship or Partner definition.

The screenshot shows a dialog box titled "Partner Relationship: Example X12 Relationship" with a blue header bar. It has five tabs: "General", "Sending Partner" (selected), "Recipient Partner", "Trade Agreements", and "Acknowledgments".

Under the "Sending Partner" tab, the following fields are visible:

- Sender Name: Example X12 Sender (dropdown menu)
- Location: X12-SEND-LOC (text field)
- Interchange section:
 - ID: ICH-SEND-ID (text field)
 - Qual: ZZ (text field)
 - Int ID: (empty text field)
 - Int ID2: (empty text field)
- Functional Group section:
 - ID: FG-SEND-ID (text field)
 - Qual: (empty text field)
 - Int ID: (empty text field)
 - Int ID2: (empty text field)

At the bottom left of the dialog is a "More..." button. At the bottom right are three buttons: "OK", "Cancel", and "Apply".

Partner Relationship: Example X12 Relationship

General | Sending Partner | **Recipient Partner** | Trade Agreements | Acknowledgments

Recipient Name: Example X12 Recipient

Location: X12-REC-LOC

Interchange

ID: ICH-REC-ID Qual: ZZ

Int ID: Int ID2:

Functional Group

ID: FG-REC-ID Qual:

Int ID: Int ID2:

More...

OK Cancel Apply

Partner: Example X12 Sender

General | **IDs** | Trade Agreements | Acknowledgments | Groups | Alias

Interchange

ID: ICH-SEND-ID Qual: ZZ

Int ID:

Int ID2:

Functional Group

ID: FG-SEND-ID Qual:

Int ID:

Int ID2:

More...

OK Cancel Apply

The image shows a software dialog box titled "Partner: Example X12 Recipient". It has a blue header bar with a small icon on the left and standard window controls (minimize, maximize, close) on the right. Below the header is a tabbed interface with five tabs: "General", "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "IDs" tab is currently selected and highlighted. The main area of the dialog is divided into two sections: "Interchange" and "Functional Group". Each section contains three input fields: "ID:", "Int ID:", and "Int ID2:". In the "Interchange" section, the "ID:" field contains the text "ICH-REC-ID" and the "Qual:" field contains "ZZ". The "Functional Group" section has the "ID:" field containing "FG-REC-ID" and the "Qual:" field is empty. At the bottom left of the dialog is a "More..." button. At the bottom right are three buttons: "OK", "Cancel", and "Apply".

Partner: Example X12 Recipient

General IDs Trade Agreements Acknowledgments Groups Alias

Interchange

ID: ICH-REC-ID Qual: ZZ

Int ID:

Int ID2:

Functional Group

ID: FG-REC-ID Qual:

Int ID:

Int ID2:

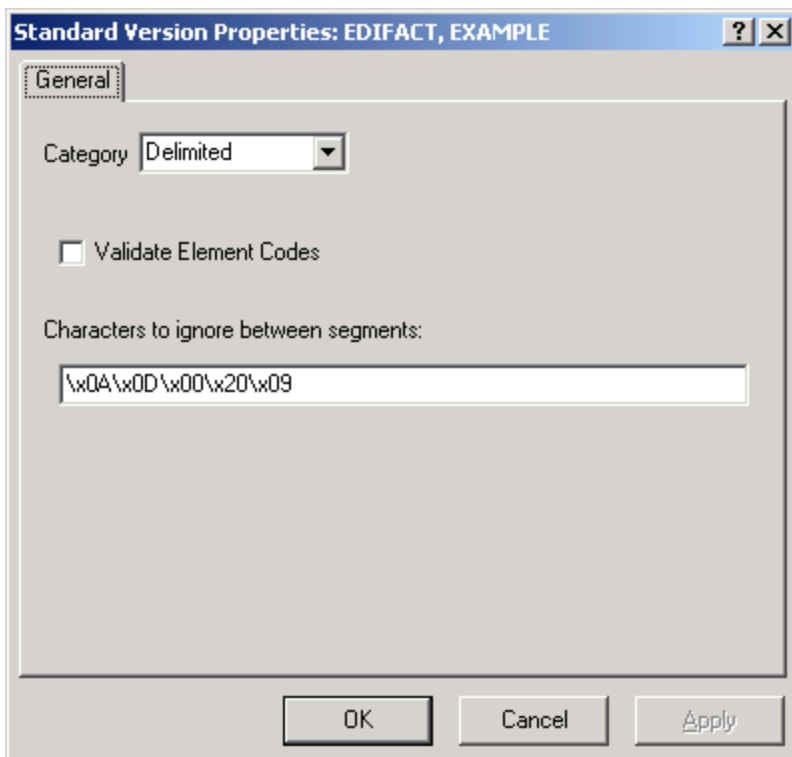
More...

OK Cancel Apply

Step 14. Generate (Validation Only or Translation)

Each interchange is routed as a separate file. How the information is grouped to be routed depends on settings in the Trade Agreement Profile for **Action** and **Error Action**, and on the creation of the output when you are translating.

- 1 Determine if the standard for this outbound wrapper is delimited or fixed-length by looking at the value for **Category** (Standard Version Properties window).



- a) If the standard version is delimited, use the generation routine for delimited data, and use the service character values in Trade Agreement Options window, **Misc** tab.

The screenshot shows a dialog box titled "Partner Trade Agreement Options: POTOORDERS" with a standard Windows window control bar (minimize, maximize, close) and a help icon. The "Output:" field is set to "1,EDIFACT,Example,ORDERS" and shows "1 of 1". Below this are three tabs: "Sending Partner", "Recipient Partner", and "Misc", with "Misc" being the active tab. The "Output Fields" section contains two checkboxes: "Request Acknowledgment" and "Test", both of which are unchecked. The "Service Characters" section contains six input fields: "Segment Terminator", "Component Delimiter" (containing "@"), "Release Character", "Tag Delimiter", "Element Delimiter", and "Decimal Mark". The "Repeat Separator" field is also present but empty. At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Apply".

- b) Where values are missing on the **Misc** tab of the Trade Agreement Properties window, then use the values in the Wrapper Properties window, **Service Characters** tab for the output wrapper. Use the offset values to find the service characters from the incoming document first. When the offset values are blank, use the specified values.

The screenshot shows the 'Wrapper Properties: EDIFACT,2,UNBB' dialog box with the 'Service Characters' tab selected. The dialog contains a table with two columns: 'Value' and 'Offset'. The rows are: Segment Terminator (Value: \x1C, Offset: blank), Tag Delimiter (Value: \x1D, Offset: blank), Element Delimiter (Value: \x1D, Offset: blank), Component Delimiter (Value: \x1F, Offset: blank), Repetition Separator (Value: blank, Offset: blank), Release Character (Value: blank, Offset: blank), and Decimal Mark (Value: ., Offset: blank). At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

	Value	Offset
Segment Terminator	\x1C	
Tag Delimiter	\x1D	
Element Delimiter	\x1D	
Component Delimiter	\x1F	
Repetition Separator		
Release Character		
Decimal Mark	.	

- c) If the standard is fixed, SWIFT or XML, use the generation routine for the appropriate type of data.

The screenshot shows a dropdown menu with the following options: Delimited, Fixed (highlighted), SWIFT, and XML.


- 2 Find the wrapper set definition identified on the trade agreement to generate the wrapper data.

Trade Agreement Output Properties

General | Summary Document | Alias


Sequence: 1 Output #: 1

Document Map:

 [EXAMPLE-X850](#)

Standard: EXAMPLE
Version: 1
Doc Id: [FPO](#)

Wrapper Map:

 [EXAMPLE-FWRAP](#)

Standard: EXAMPLE
Version: 1
Wrap Id: [FWRAP](#)

OK Cancel Apply

Wrapper: EXAMPLE,1,FWRAP

Description:

Header Segments:

Lvl	Seq	Seg Tag	Description	Rqmt
▶ 1	10	***	Fixed Header Record	M

Contents:

Header: [Purchase Order Header](#)

Trailer:

Trailer Segments:

Lvl	Seq	Seg Tag	Description	Rqmt
▶				

- 3 Generate the output header wrapper segment (the remaining header wrapper segments will be generated when the TRM encounters them as it sequentially processes the interchange, until it reaches the first document header).

4 Generate the document segments.

Document: EXAMPLE,1,FPO

Description: Functional Group:

L/S	Level	Area	Seq	Tag	Description	Rqmt	Max Occ
S	0	1	10	HDR	Purchase Order Header	M	1
S	0	1	20	BUY	Buyer	O	1
S	0	1	30	SEL	Seller	O	1
S	0	2	40	DTL	Line Item Details	M	999999
S	0	3	100	SUM	Summary totals	O	1

5 Create a summary document, if required.

Trade Agreement Output Properties

General Summary Document Alias

Summary Doc Map:

Standard:

Version:

Doc Id:

Security User Exit:

OK Cancel Apply

6 Generate the wrapper trailer segments.

NOTE: When the TRM does contents validation only or translation, it routes the entire interchange. If the **Error Action** is set to **Reject** (Trade Agreement Profile window, **Options** tab) and there is an error in the document, it deletes rejected documents from the outgoing interchange, adjusting trailer counts as necessary. When there are errors in the interchange or functional group level wrappers, the TRM rejects the entire interchange.

Step 15. Determine Routing (Validation)

When the TRM is ready to route a file that does not require translation, it obtains the routing address from the location it found for the recipient of the last wrapper segment it parsed for the inbound wrapper.

Step 16. Determine Routing (Translation)

The TRM must find both source and destination locations for the output or it terminates processing. The search for locations for outbound information is the same search described in a previous step, determining IDs for outbound partners, with the following results:

- Use the location for the outbound sending partner as the source location.
- Use the location for the outbound recipient partner as the destination location.

You may want to override the location defined for the sender or recipient. You can provide alternative locations in one of four ways:

- Use the location associated with the outbound partners specified on the Trade Agreement Options window for a trade agreement associated with a partner relationship, partner, or standard ID.
– or –
- Define an override location on the Trade Agreement Options window for a trade agreement associated with a partner relationship, partner, or standard ID.
– or –
- Use an alias type on the Trade Agreement Profile that matches an alias for a particular partner.
– or –
- Override the previous 3 by mapping the location value using Edibasic **SetField()** statements in the document map **Condition** method.

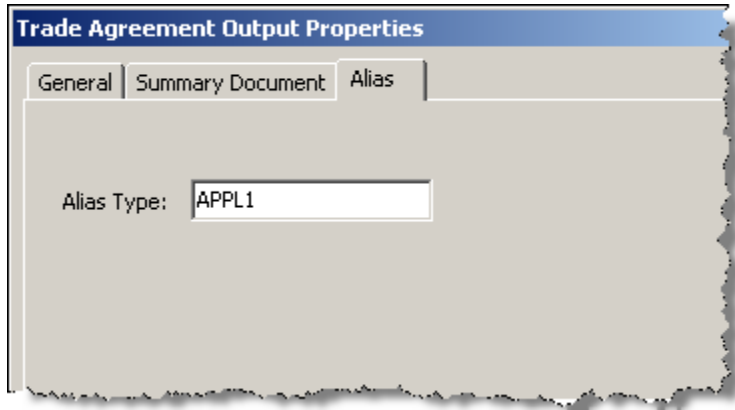
Once it has checked for locations for the outbound sending and recipient partners, the TRM checks the configurations for overrides or for missing source and destination locations as follows:

- 1 Using the trade agreement profile it found earlier, search for an override location on the Trade Agreement Options for other options, such as:
 - An alternative destination location that will override the location associated with the identified partner.
 - Service characters that will override whatever may have been associated with an outbound delimited wrapper.
 - Test and acknowledgment flags.

If destination routing information is still missing, proceed to the next option, to use aliases.

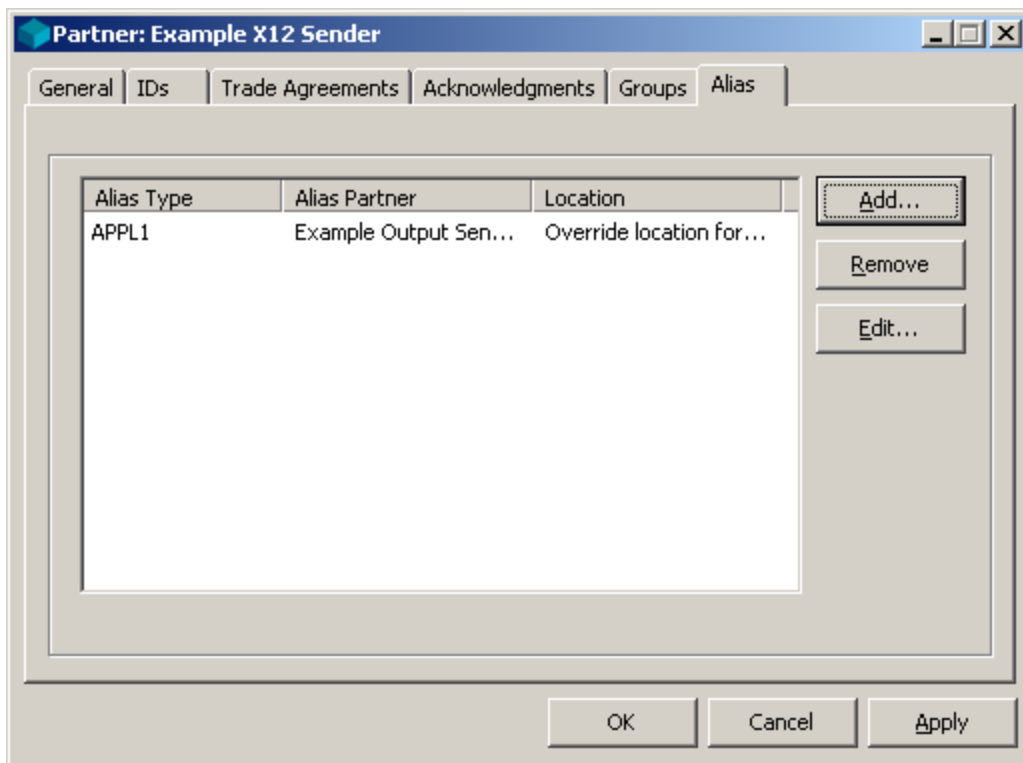
- 2 Search for any missing routing information using partner aliases as follows:

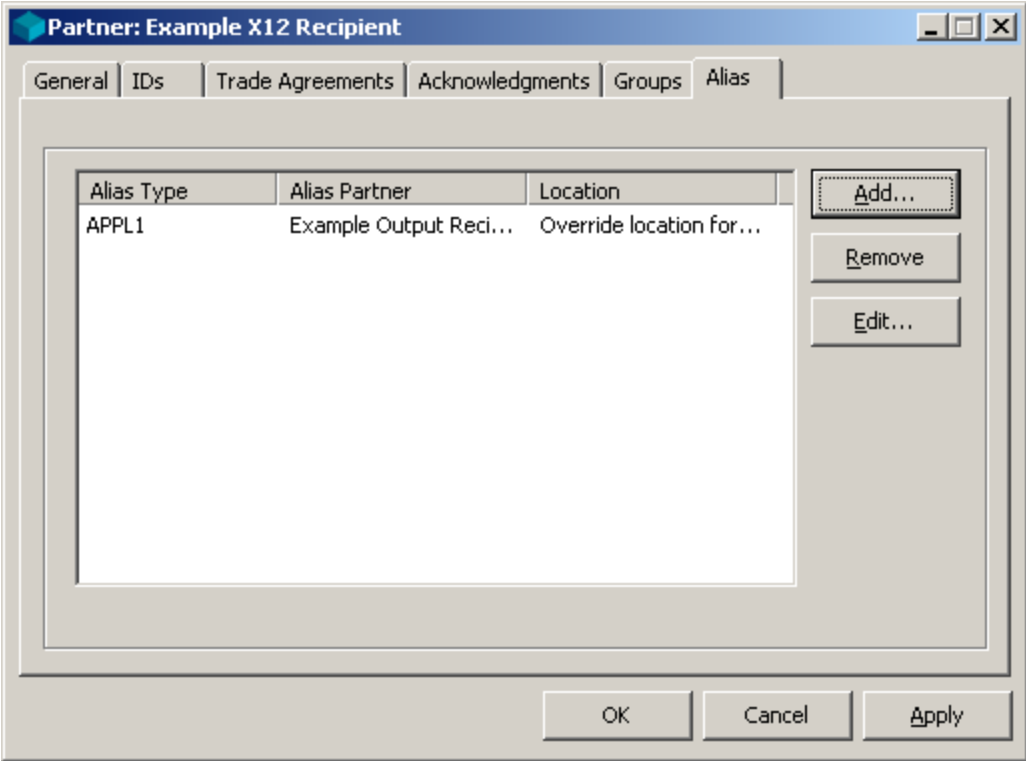
- a) Determine a recipient alias type for the alias type defined in the Trade Agreement Output Options window, **Alias** tab.



- If the alias type contains the character *R*, then replace the *R* to determine the recipient alias type.
 - otherwise –
 - If the alias type does not contain the character *R*, then use the alias type unchanged for the recipient alias type.
- b) Find the outbound recipient partner.

- If an inbound recipient partner exists, search for the recipient alias type in the inbound recipient alias list.
 - otherwise –
- If an inbound recipient partner does not exist, and if an inbound sender exists, search for the recipient alias type in the inbound sender alias list.
 - otherwise –
- If no alias is found, proceed to the next option, locations on inbound definitions.





- 3 If routing information is still missing, look for locations from the inbound definitions, which may mean a Partner Relationship or Partner definitions for the recipient or sending partner.

Partner Relationship: Example X12 Relationship

General | **Sending Partner** | Recipient Partner | Trade Agreements | Acknowledgments

Sender Name: Example X12 Sender

Location: **X12-SEND-LOC**

Interchange

ID: ICH-SEND-ID Qual: ZZ

Int ID: Int ID2:

Functional Group

ID: FG-SEND-ID Qual:

Int ID: Int ID2:

Partner Relationship: Example X12 Relationship

General | Sending Partner | **Recipient Partner** | Trade Agreements | Acknowledgments

Recipient Name: Example X12 Recipient

Location: **X12-REC-LOC**

Interchange

ID: ICH-REC-ID Qual: ZZ

Int ID: Int ID2:

Functional Group:

- 4 If the inbound recipient partner location is missing, use Standard ID routing to determine the destination location. First it checks **Data field** values, then **Source** values and finally, as a last resort, **Default values**, using the first one that produces a destination location.
- Use one of the input partner IDs associated with the recipient partner when:
 - Data field** is checked together with **Partner ID** or **Partner Int ID**
 - and –
 - Use Source before Data Field** is not checked
 - Use the inbound, source location when:

- **Source** is checked and **Data field** is not checked
 - or –
 - When **Data field** is checked and **Use Source before Data Field** is also checked
- c) Use the default destination location.

Standard ID: X850TEST, 1, X12, EXAMPLE, ISA

General | Partners | **Routing** | Trade Agreements | Acknowledgments

Location: X850TEST
 Standard: [X12](#) Version: [EXAMPLE](#) Wrapper ID: [ISA](#)

Default Output Routing

Data Field: Partner ID Source Address Use Source before Data Field
 Partner Int ID Use Source before Partner Location

Default: Source: _____
 Destination: _____

Default Acknowledgment Routing

Data Field: Partner ID Source Address Use Source before Data Field
 Partner Int ID Use Source before Partner Location

Default: Source: _____
 Destination: _____

OK Cancel Apply

- 5** Use standard ID routing to determine the source location, using whichever produces a location first, when:
- a) **Use Source before Partner Location** is checked and when a source location has already been found on the partner definitions. This overrides a partner-based source location.
 - b) **Default** is checked and **Source** contains a value.
- 6** If the TRM does not find both a source and destination location, it stops processing.

Step 17. Repeat Steps 14 And 16 for Each Defined Output (Translation)

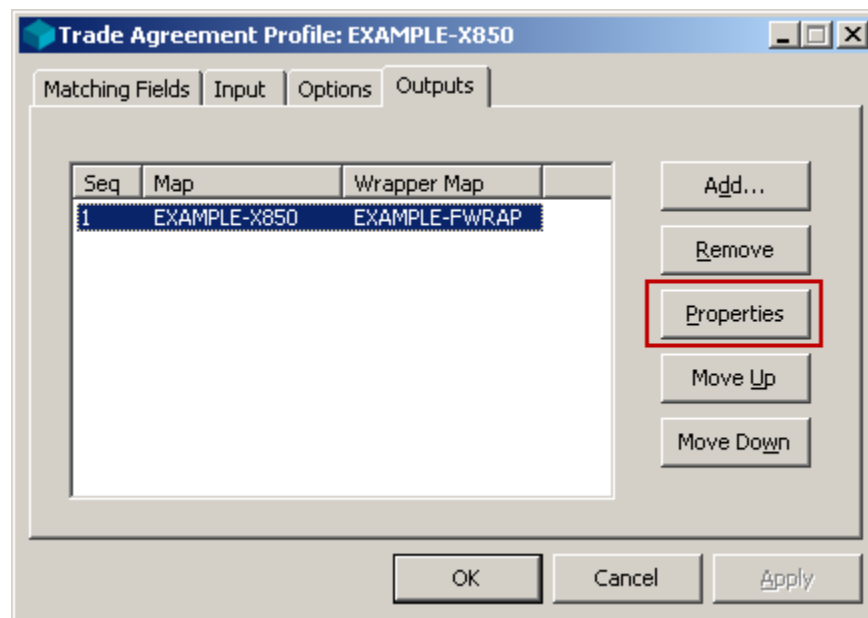
Since there can be multiple outputs from a single trade agreement definition, the TRM must repeat the steps to find outbound partners, determine routing, and generate and route the data for each output (Trade Agreement Profile window, **Outputs** page).

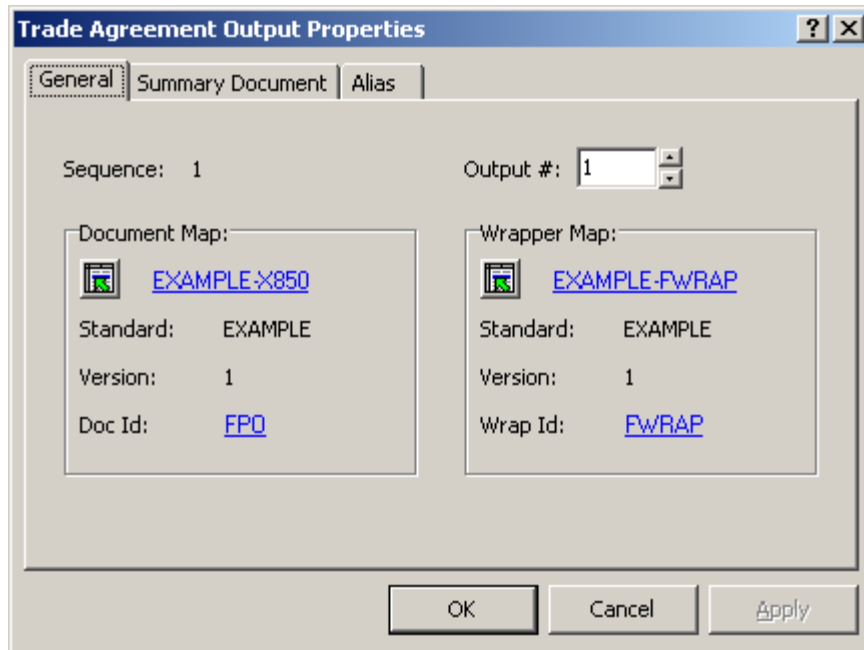
The TRM processes documents sequentially, and will break when it encounters any of the following conditions for the output.

NOTE: If the appropriate level of wrapper or partner IDs are not defined for the output file, the TRM decides to break the output based on the header levels and partner IDs defined for the input file.

- A new outbound interchange-level header will force a file break, beginning a new output stream (there will always be only one interchange per outbound file).
- A new outbound location will force a file break, beginning a new output stream.
- The Document Level Break field on the Trade Agreement Options window, **Misc** tab will force the current document to be placed in its own interchange and file, beginning another output stream thereafter.
- A new functional group-level ID will force a break at the functional group level within the same output stream.

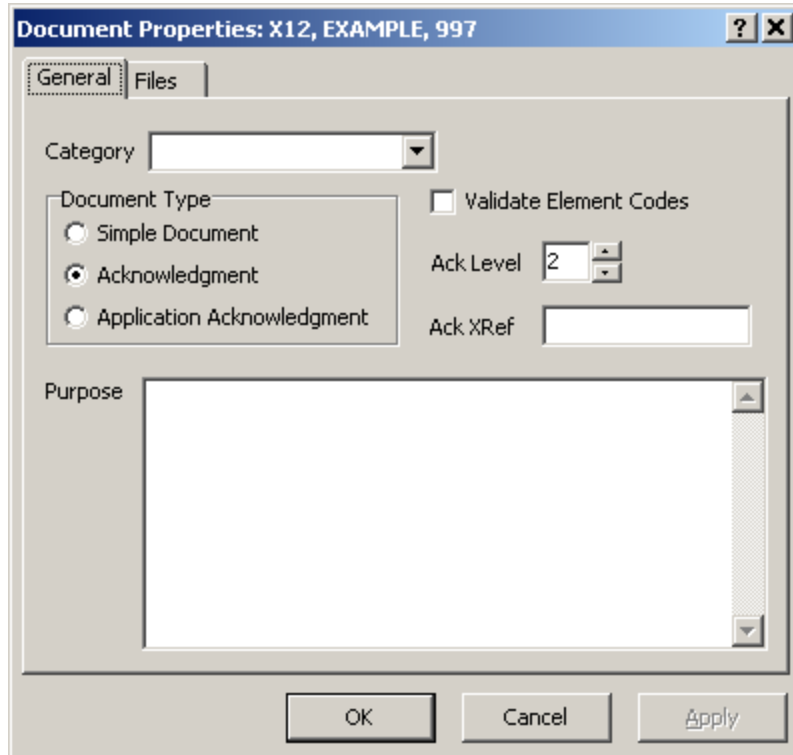
NOTE: You can group output documents by assigning an output number on the Trade Agreement Output Properties window for any output specified for the entire interchange, even for multiple outputs that have different trade agreement properties. However, such grouping is subject to the list of breaking conditions, which might interrupt the grouping.





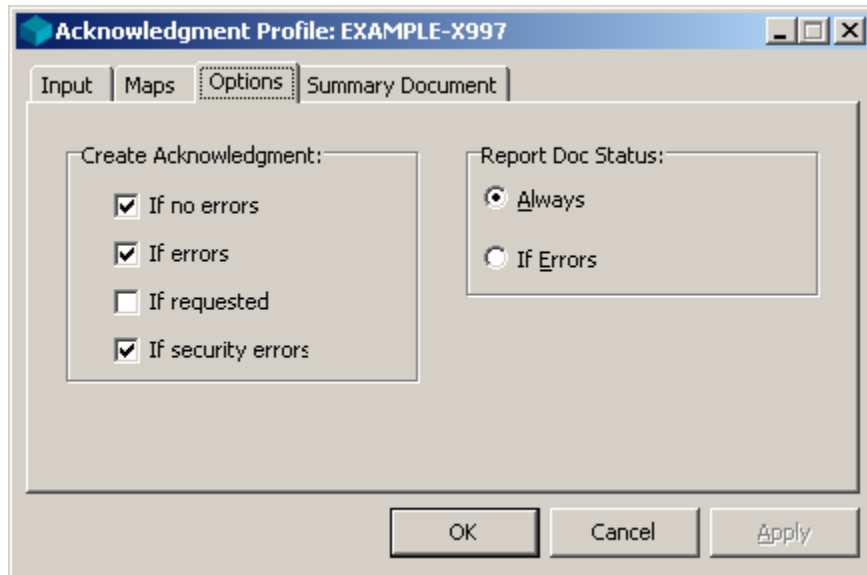
Step 18. Create Backward Acknowledgments, As Required

When the TRM finds an acknowledgment profile, it will use that information to create acknowledgments at the level identified on the Document Properties window, General page. This might be one for every interchange (level 1) or one for every functional group within the interchange (level 2). Note that the **Ack Level** box is visible only for documents of the **Acknowledgment** type.

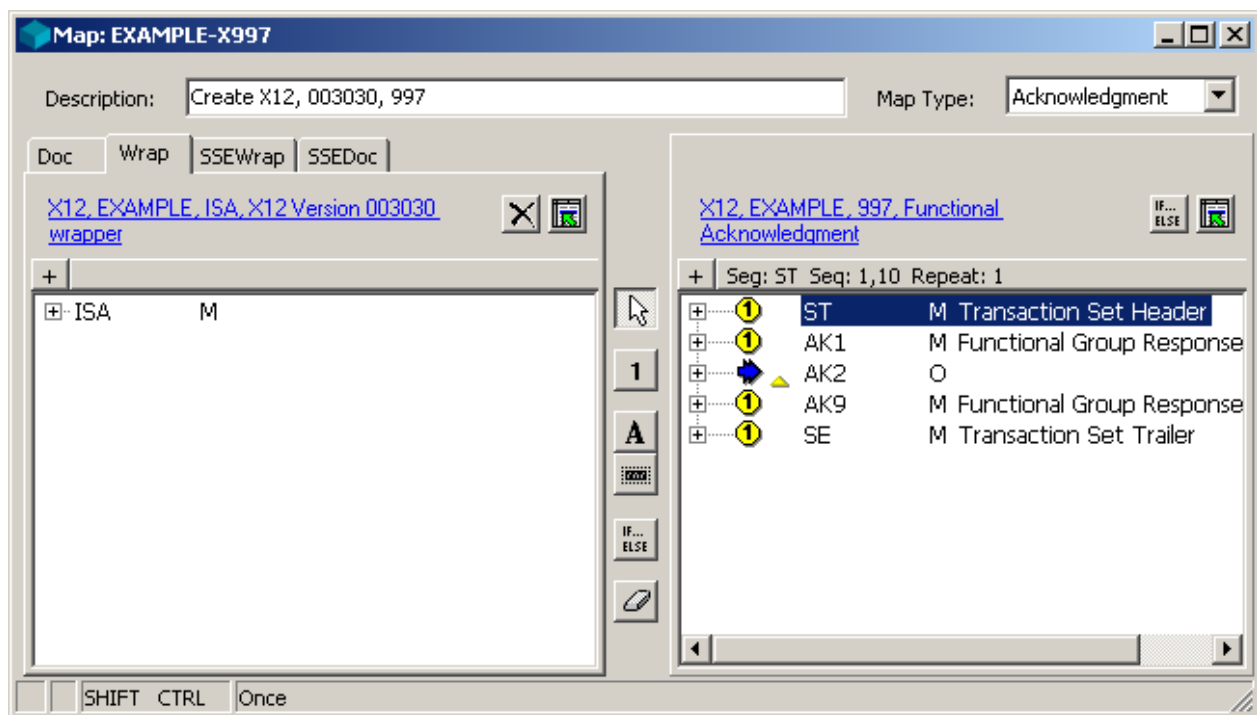
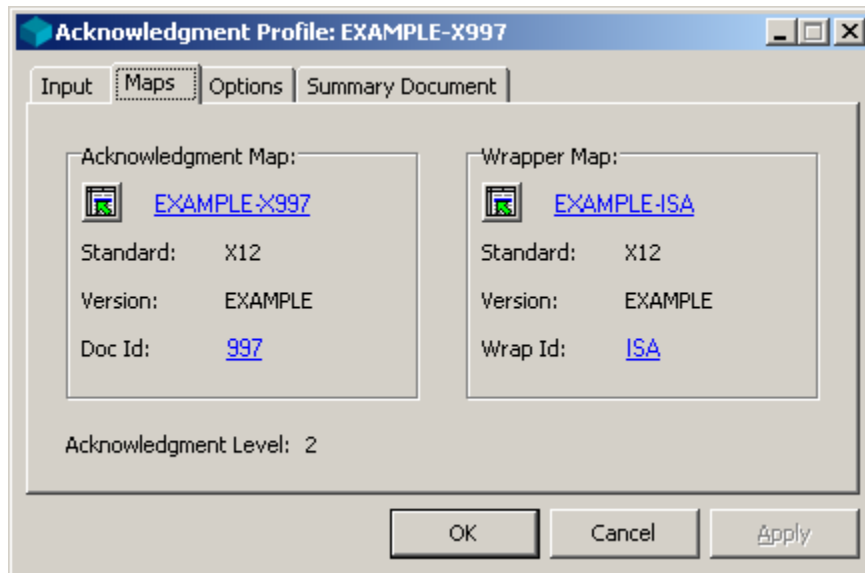


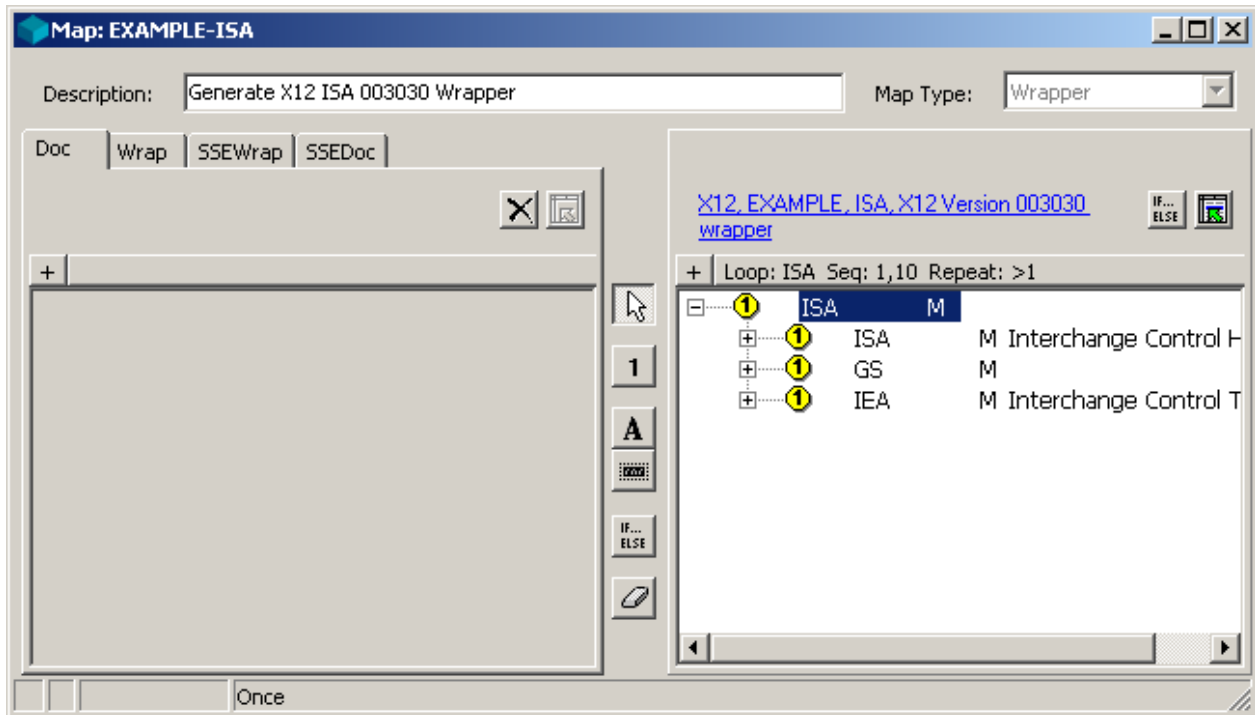
The screenshot shows the "Document Properties: X12, EXAMPLE, 997" dialog box. It has two tabs: "General" and "Files". The "General" tab is selected. The "Category" field is a dropdown menu. The "Document Type" section contains three radio buttons: "Simple Document", "Acknowledgment" (which is selected), and "Application Acknowledgment". To the right of these is a checkbox for "Validate Element Codes" which is unchecked. Below that is a spinner box for "Ack Level" set to "2", and an empty text box for "Ack XRef". At the bottom is a large "Purpose" text area. At the very bottom are three buttons: "OK", "Cancel", and "Apply".

Whether the TRM creates an acknowledgment depends on the selections in the **Create Acknowledgment** box. Whether or not it reports document status information depends on what is selected in the **Report Doc Status** box.



It uses the maps defined on the Acknowledgment Profile window, **Maps** tab to create the acknowledgments.





The processing report indicates if acknowledgments were generated. Notice that there is now a second output stream created. The previous documents were all grouped in one output file.

```
>>> Wrapper set definition file name:
C:\MWTranslator\DefaultEnv\Cfg\x12\W-X12-EXAMPLE-ISA.txt <<<
>>> Map file name: C:\MWTranslator\DefaultEnv\Cfg\map\M-EXAMPLE-ISA.txt <<<
>>> Document definition file name:
C:\MWTranslator\DefaultEnv\Cfg\x12\D-X12-EXAMPLE-997.txt <<<
>>> Map file name: C:\MWTranslator\DefaultEnv\Cfg\map\M-EXAMPLE-X997.txt <<<
Acknowledgment Profile found on Partner Relationship: EXAMPLE-X997
Functional Acknowledgment Output stream created:
C:\MWTranslator\DefaultEnv\Out\BU08MU1.txt
Output number: 2 Source: X12-REC-LOC Destination: X12-SEND-LOC

>>> Control reference file name: C:\MWTranslator\DefaultEnv\Cfg\CTRLREF.TXT <<<
Wrapper level 1 Status: A Control Reference: 000010001
```

Step 19. Determine Routing for Backward Acknowledgments

The TRM always uses the same IDs it received in the incoming wrappers for backward acknowledgments, reversing their functions. So the sender IDs of the inbound document become the recipient IDs of the acknowledgment, and the recipient IDs of the incoming document become the sender IDs of the acknowledgment. The TRM determines the routing locations, both source and destination, for acknowledgments as follows:

- 1 Using the acknowledgment profile it found earlier, the TRM searches the Acknowledgment Options for a source and destination location. If there is a value in the **To** field of the **Send Outputs** box, it routes the acknowledgment to this location. The source location will come from here also, if there is something in the **From** field.

Partner Relationship Acknowledgment Options: EXAMPLE-X997

General

Send Outputs:

From:

To:

Service Characters

Copy from input stream

Segment Terminator Component Delimiter Release Character

Tag Delimiter Element Delimiter Decimal Mark

Repeat Separator

OK Cancel Apply

However, these values, and any other partner values that the TRM might find, will be overridden if the **Use Source before Partner Location** is selected in the **Default Acknowledgment Routing** box on the **Routing** tab of the Standard ID window.

The screenshot shows a window titled "Standard ID: X850TEST, 1, X12, EXAMPLE, ISA" with tabs for General, Partners, Routing, Trade Agreements, and Acknowledgments. The Routing tab is active. The window displays the following configuration:

Location: X850TEST
Standard: [X12](#) Version: [EXAMPLE](#) Wrapper ID: [ISA](#)

Default Output Routing

- Data Field: Partner ID [Source Address](#) Use Source before Data Field
- Partner Int ID Use Source before Partner Location
- Default: Source:
- Destination:

Default Acknowledgment Routing

- Data Field: Partner ID Source Address Use Source before Data Field
- Partner Int ID Use Source before Partner Location
- Default: Source:
- Destination:

Buttons: OK, Cancel, Apply

- 2 If there is no information in the **To** field of the **Send Outputs** box of the Acknowledgment Properties window, the TRM checks the location on the **Sending Partner** tab for a Partner Relationship, or the location on the **General** tab for a Partner definition for the sender of the original incoming document.

The screenshot shows a window titled "Partner Relationship: Example X12 Relationship". It has several tabs: "General", "Sending Partner" (selected), "Recipient Partner", "Trade Agreements", and "Acknowledgments". The "Sending Partner" tab contains the following fields:

- Sender Name: Example X12 Sender
- Location: X12-SEND-LOC (highlighted with a red box)
- Interchange section:
 - ID: ICH-SEND-ID
 - Qual: ZZ
 - Int ID: (empty)
 - Int ID2: (empty)
- Functional Group section:
 - ID: FG-SEND-ID
 - Qual: (empty)
 - Int ID: (empty)
 - Int ID2: (empty)

The screenshot shows a window titled "Partner: Example X12 Sender". It has several tabs: "General" (selected), "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "General" tab contains the following fields:

- Partner Name: Example X12 Sender
- Location: X12-SEND-LOC (highlighted with a red box)

- 3 If it still has no location, search the **Routing** tab information of the Standard ID wrapper identified during parsing of the original document. Use the rules identified in the **Default Acknowledgment Routing** box, searching left to right and top to bottom. If **Use Source before Partner Location** is checked, this will override any other values for the source location, even the one listed in the **Default Source** box.

Standard ID: X850TEST, 1, X12, EXAMPLE, ISA

General Partners Routing Trade Agreements Acknowledgments

Location: X850TEST
 Standard: X12 Version: EXAMPLE Wrapper ID: ISA

Default Output Routing

Data Field: Partner ID Source Address Use Source before Data Field
 Partner Int ID Use Source before Partner Location

Default: Source:
 Destination:

Default Acknowledgment Routing

Data Field: Partner ID Source Address Use Source before Data Field
 Partner Int ID Use Source before Partner Location

Default: Source:
 Destination:

OK Cancel Apply

If the TRM finds locations for routing, it specifies these on the processing report.

It also specifies the end of translation date and time.

```
>>>> End of Translation 2009/08/03 15:53:52 <<<<
```

Map Development Guidelines

This section provides hints and techniques about mapping in the order that most map developers use when developing a new translation. It includes the following sections:

- **Use of EDI Standards:** suggestions for the most effective use of the X12 and EDIFACT standards provided with the MW Translator Workbench.
- **Review of Proprietary Formats:** questions to be answered when reviewing the characteristics of a proprietary format.
- **Definition of Proprietary Formats:** a procedure to define an MW Translator document and wrapper for a proprietary format.
- **Standard Identification:** a procedure to define a standard identification rule and a trade agreement profile that can be used for testing during the map development process.
- **Wrapper Map:** a procedure for the development of a wrapper map for a proprietary format.
- **Document Map:** general guidelines for the development of a document map.
- **Partners and Routing:** guidelines for the appropriate use of partner records to provide partner identifiers in translation output and to ensure correct routing of output messages.
- **Software Release:** suggestions for the preparation of a software release package that can be used to install a new translation into a MessageWay environment.

Use of EDI Standards

Here are some basic steps to begin:

- For an X12 or EDIFACT document, *load the appropriate standard directory into your development MW Translator database* (on page 507).
- If the document is the translation source and you require changes to comply with an implementation guide, *copy the document to a custom standard version and make changes to the copy* (on page 496).
- If the document is the translation target and you must map data from various segments or loops in a source document to provide multiple occurrences of a repeating segment or loop in a single target document, use a repeated definition of the target segment or loop. Copy the document to a custom standard version for output use only and make changes to the copy.

IMPORTANT: When you want to create one-off versions of public standards, make sure your corporate policy allows you to modify the standard.

- If the document is the translation source, and an existing translation uses the same document as input, decide how to make a distinction between the two translations:
 - **For a default translation where messages come from a particular source**, use a special source location for standard identification and invoke the trade agreement profile from the standard identification rule.
 - **For a translation specific to a recipient**, use the <Default> location for standard identification and invoke the trade agreement profile from recipient partner records.
 - **For a translation that's specific to a sending partner and a recipient partner**, use the <Default> location for standard identification and invoke the trade agreement profile from a partner relationship.
- If the document is the translation source and there's no other existing translation that uses the same document as input, decide how to invoke the trade agreement profile:
 - **For a default translation**, use the <Default> location for standard identification and invoke the trade agreement profile from the standard identification rule.
 - **For a default translation where messages come from a particular source**, use a special source location for standard identification and invoke the trade agreement profile from the standard identification rule.
 - **For a translation from a specific recipient**, use the <Default> location for standard identification and invoke the trade agreement profile from recipient partner records.
 - **For a translation that's specific to a sending partner and a recipient partner**, use the <Default> location for standard identification and invoke the trade agreement profile from partner relationships.

Review of Proprietary Formats

- 1 Review a specification of the proprietary format. *Is it delimited or fixed-format* (on page 651)?
- 2 Does the format have some form of physical record structure?

In general, delimited standards have no physical record structure and fixed-format standards have physical record structure. A text file has record structure by virtue of the new line character at the end of each record.
- 3 If the format is delimited, is there an appropriate segment tag at the start of each logical record?

If not, a pre-processing user exit might be needed.
- 4 If the format is fixed-format, is there *an appropriate segment tag somewhere within each logical record* (on page 630)?

If not, a pre-processing user exit might be needed.
- 5 What are the *data types needed for the elements* (on page 646) within the format?
- 6 What is the *character set* (on page 644) for each data type?

7 Are logical records in a predictable sequence?

If they're not, a pre-processing user exit might be needed to sort records.

8 Is there a need for an MW Translator wrapper definition?

- Are there file header records?
- Are there file trailer records?
- Does the file contain multiple batches, each with a batch header and possibly with a batch trailer?
- Is there a record that provides a partner identifier? (Only elements in wrapper segments can be associated with MW Translator partner fields.)
- Is a pre-processing user exit needed to create a suitable file header or batch header?
- If there's no requirement to define wrapper segments, you must *define a null wrapper* (on page 146).

9 Does the file contain a single document or multiple documents?

10 Is there a mandatory record that can be used as a document header?

If not, a pre-processing user exit might be needed to create document headers.

11 Which records are mandatory within a document?

12 Do some records have multiple occurrences, that is, repeating segments?

13 Do some groups of records have multiple occurrences, that is, loops or groups?

14 Does the document have a hierarchical structure, that is, nested segments and loops/groups?

15 If the document is the translation target and you must map data from various segments or loops in the source document to provide multiple occurrences of a repeating segment or loop single in a target document, then you can use a repeated definition of the target segment or loop to simplify mapping.

16 If the document is the translation source, decide how to invoke the trade agreement profile:

- **For a default translation**, use the <Default> location for standard identification and invoke the trade agreement profile from the standard identification rule.
- **For a default translation for messages from a particular source**, use a special source location for standard identification and invoke the trade agreement profile from the standard identification rule.
- **For a partner-specific translation when the input contains a recipient partner ID**, use the <Default> location for standard identification and invoke the trade agreement profile from recipient partner records.
- **For a partner-specific translation when the input contains both a sending partner ID and a recipient partner ID**, use the <Default> location for standard identification and invoke the trade agreement profile from partner relationships.

Definition of Proprietary Formats

- 1 In Data Explorer, *add a standard for the proprietary format* (on page 645) and then *add a version* (on page 645) to the standard.
- 2 In the standard version, *define the necessary data types* (on page 646).
- 3 *Define the properties of the standard version* (on page 651).
- 4 *Use a top-down approach to add a document definition to the standard version* (on page 120):
 - Add the document and define the sequence and nesting of segments.
 - Double click each segment and define the sequence of elements within the segment.
 - Double click each element and define its attributes.
- 5 *Define the properties of the document* (on page 123).
- 6 *Use a top-down approach to add a wrapper definition to the standard version* (on page 127):
 - Add the wrapper and define the sequence of header and trailer segments. (A null wrapper has no header and trailer segments.)
 - Specify the name of the document header segment. (Mandatory for all wrappers, including null wrappers.)
 - If there is one, specify the name of the document trailer segment.
 - Double click each segment and define the sequence of elements within the segment.
 - Double click each element and define its attributes.
- 7 Where appropriate, *associate wrapper elements with MW Translator fields* (on page 136).
- 8 *Define the properties of the wrapper* (on page 130).

Standard Identification

- 1 In Partner Explorer, *add a standard identification rule* (on page 858) in accordance with your earlier decisions to identify the input wrapper uniquely.
- 2 *Define a trade agreement profile* (on page 222) to validate the translation input.
- 3 *Invoke the trade agreement profile* (on page 859) in accordance with your earlier decisions. (If necessary, create test partner definitions and partner relationships.)
- 4 Run tests to confirm that you can successfully read and parse the test file. As needed, make adjustments to your test file, document definition, wrapper definition or standard identification rule to ensure success before proceeding to mapping.

Wrapper Map

- 1 If your translation target is an X12 or EDIFACT standard, you do not need to create a wrapper map. Use the appropriate wrapper map that was loaded with the standard.

NOTE: If you have created a copy of the standard, you will have to change the target wrapper name in the wrapper map.

- 2 If your translation target is the proprietary format, in the data explorer add a wrapper map and specify your wrapper definition as the target wrapper.

If you have a null wrapper, no further work is needed.

- 3 If you need to assign values from the source wrapper to your target wrapper, much of the information is available in an MW Translator internal table accessed from the **SSEWrap** tab on the source side of the mapping window.

If this does not provide the information that you need, click the **Wrap** tab on the source side of the mapping window and select the appropriate source wrapper. For most wrapper maps, this is not necessary.

- 4 Use visual mapping techniques as much as possible to define the occurrences of wrapper segments and to assign values to wrapper elements:

- **Once** tool to generate an occurrence of a target segment
- **Literal** tool to assign a literal to a target element
- **Internal Field** tool to assign the value of an MW Translator field to a target element
- Drag-and-drop from a **Wrap** element or an **SSEWrap** element to a target element

- 5 Use the **MapEle** method for the remaining target elements.

Document Map

- 1 In Data Explorer, add a document map and select the appropriate source document and target document.
- 2 Use visual mapping tools to generate occurrences of target loops and segments:
 - **Once** tool to generate a single occurrence of a target segment
 - Drag-and-drop from a non-repeating source segment to a target segment to create a single occurrence whenever the source segment is present
 - Drag-and-drop from a repeating segment or loop to a target repeating segment or loop to create the same number of occurrences of the target as there exist for the source
- 3 If appropriate, define **Condition** methods to govern if a target segment or loop should be generated.
- 4 If the target document is fixed-format, use the **Literal** tool to assign a value to each segment tag.

- 5 Modify your trade agreement profile to specify translation with your wrapper and document map.
- 6 Test your translation to confirm that you are generating the expected segments and loops.
- 7 Define mapping rules for a small number of target elements and then test your translation.
- 8 Repeat this process until mapping rules have been defined for all target elements. This iterative process ensures that you remain focused on the last changes that you made to the map and helps ensure a defect-free translation.

Partners and Routing

- 1 Does the translation input provide a sending partner ID (and qualifier) needed in the output wrapper?
If not, you need to define a *default sending partner in the standard identification rule* (on page 308) or an *override sending partner in the options of the invocation of the trade agreement profile* (on page 881).
- 2 Does the translation input provide a recipient partner ID (and qualifier) needed in the output wrapper?
If not, you need to define a *default recipient partner in the standard identification rule* (on page 308) or an *override recipient partner in the options of the invocation of the trade agreement profile* (on page 881).
- 3 Is there a need to convert a partner identifier from that used in the input to one that's appropriate for the output?
If so, you can *use a partner alias* (on page 785) to make the conversion or you can specify an *override partner in the options of the invocation of the trade agreement profile* (on page 881).
- 4 Does your use of partner records provide an appropriate source location and destination location for the output?
If not, you can *modify partner location names* (on page 786)(make sure this does not invalidate other translations that are using the same partner records) or *provide an override location in the options of the invocation of the trade agreement profile* (on page 881).

Software Release

When you have completed development of a translation, you need to prepare a software release package that can be used to install the translation into the MessageWay environment. In some cases, the software release package might include more than one translation. The following suggestions are derived from successful practices used by the author. They can be adapted to conform to any standard practices that might exist in your development team.

- 1 Create one or more import files to contain all of the definitions needed for your translation to run successfully in an empty MW Translator Workbench database.

- 2** Create one or more test files that can be used by the installer to verify that installation of the translation into another Workbench database has been successful.
- 3** Provide test output and translation reports that you have obtained from the test files for use in installation verification.
- 4** Write a software release document that describes the contents of the release and that provides a detailed procedure to install and verify the release in a Workbench environment and in a MessageWay environment.
- 5** Test the installation and verification procedure.
- 6** If your translation requires specific configuration of partner records to be successful, write a user manual to provide the detailed procedure.
- 7** Test the partner configuration procedure.
- 8** Ensure that the software release package is stored in a folder on a suitable server for future use.

This page intentionally blank

Defining Standards

Overview

Remember that when you transfer business information you must specify:

- The structure of the input data
- The structure of the output data

For each structure, you have an optional envelope (wrappers) and the content (documents). Both wrappers and documents comprise segments, and segments comprise elements.

Typically, in this environment, one of the data structures is a public standard, which is supplied with the Workbench and which you can load from compressed disk files as required. You must define any data structures not provided with the Workbench.

Defining Standards

Standards provide rules for structuring your data and grouping structures to send in electronic files. The term standards refers to these data structures, because they are based on public national and international standards for sending business documents from application to application. This standardization of business information was a necessary step to make the exchange of business information more efficient. It required several years where businesses devoted technical and business information expertise to arrive at a consensus to develop public standards. The process continues with new versions of standards and new standards appearing every year; definitions may be added, deleted or changed. Entire new structures may be created and structures may change as businesses respond to quickly changing information needs. If there is anything that everyone has learned, it is that standardization is essential and it is constantly evolving.

Certain public national (X12) and international (EDIFACT) standard definitions are included with the Workbench. These standard definitions are installed in compressed files for you to load into your Workbench databases as needed. You can define other data structures using Data Explorer.

Task 1. Creating a Proprietary Standard

Within our terminology, a proprietary standard is any data structure that is not an exact copy of the public standards created by standards organizations. This only means that users are responsible for entering any other structures themselves. You can create the same type of definitions for all standards, whether they are public or proprietary.

For example, some trading partners may require that you trade with them using a modified version of the public standard. Under such circumstances, you can easily define a special version of a public standard using the copy utility and then make changes as required.

The work of creating your proprietary standard may not be this simple. It is easy entering definitions. It can take time making the decisions that result in a document structure that you want.

Typically, this may be one of the most time-consuming tasks, because often you do not have a business document that your application creates that will satisfy your current EDI need. If there is no current document that you can use as input, you will have to work with your applications group to determine how to generate a file with the information you need. Then you should agree on a document layout. This is not a trivial task. Hidden in it are two major issues:

- What information you need to pass to and from your application to meet your needs and those of your partner's, and
- What data format you can design that will allow you to minimize data manipulation in your map and possibly in other code that your application group might have to write.

With regard to the first point, your trading partner may have made these decisions for you and hand you their implementation guide, with which you must comply. An implementation guide includes specific instructions about what documents to exchange, what information to include in a document, and where to put it in the document structure. If there is no implementation guide, you are on the way to creating your own.

To minimize the work, it helps to understand EDI in general and how your industry uses a particular set of documents to transfer business information. It is also very important to have a responsive working relationship with your applications group and your trading partner. Once you have completed this hard work, the easy work begins: entering the configurations.

The sample windows you see are from the example that is delivered with the Workbench. This example shows a translation from an X12 850 (Purchase Order) to a proprietary purchase order.

For specific steps, refer to the online help, and for a complete exercise and more complete explanation of details refer to the *MW Translator Workbench Tutorial*, which is delivered on your installation medium as a PDF file.

Each time you create or change a document or wrapper that you will need to use during processing, you must generate a new text file that includes the changes. You can generate the wrapper or document with everything else at the end of your entire configuration process, but it is better to generate them after you define the entity, so you can correct any problems at an early stage.

Task Tree to Define Standards

Your tasks are as follows:

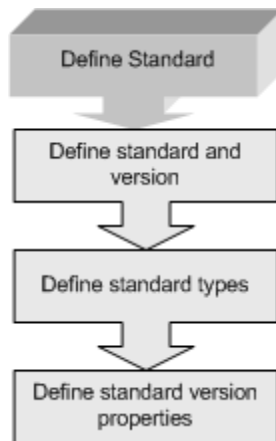


Figure 1. Defining Standards Task Tree

Task 1.1. Specifying Your Standard and Version

All data structure definitions belong to a version within a standard, which is reflected in the view of the folders within Data Explorer. This view reflects the development of public standards that may have multiple versions of a standard, for both wrappers and documents. This is less likely for proprietary standards, which tend to be simpler.

Task Tree to Define a Standard

The following task tree shows the sub-tasks to specify your standard and version.

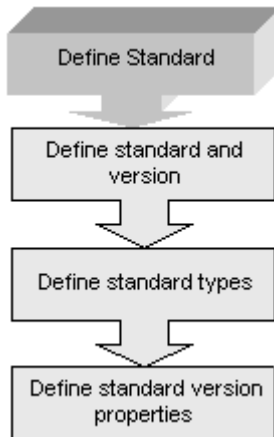
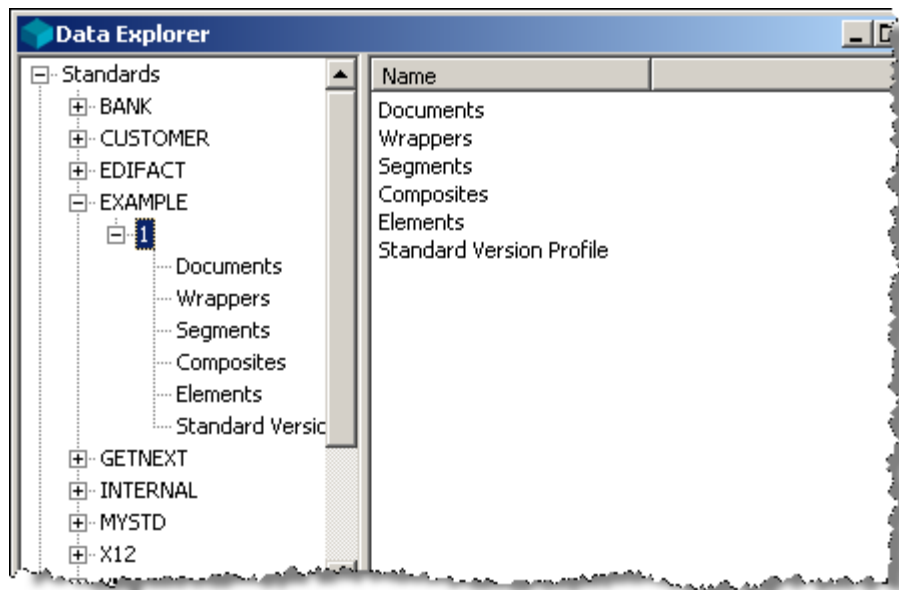


Figure 2. Sub-tasks to Specify Standard and Version (Defining Standards Task Tree)

Task 1.1.1. Defining Your Standard and Version

Your first step is to *define your standard* (on page 645) and then a *version within this standard* (on page 645). Use names that are meaningful for you. Standards are simply a way of grouping related definitions as versions. All configuration information is associated with the version of the standard, not the standard.

Note that once you have created your standard and version, the Workbench creates a set of folders for you that will allow you to access all of the definitions related to your new standard version.



The Example standard is the proprietary standard that is used with the sample translation delivered with the Workbench.

Your next step will be to define the Standard Version Profile, which will include the Standard Types and the properties.

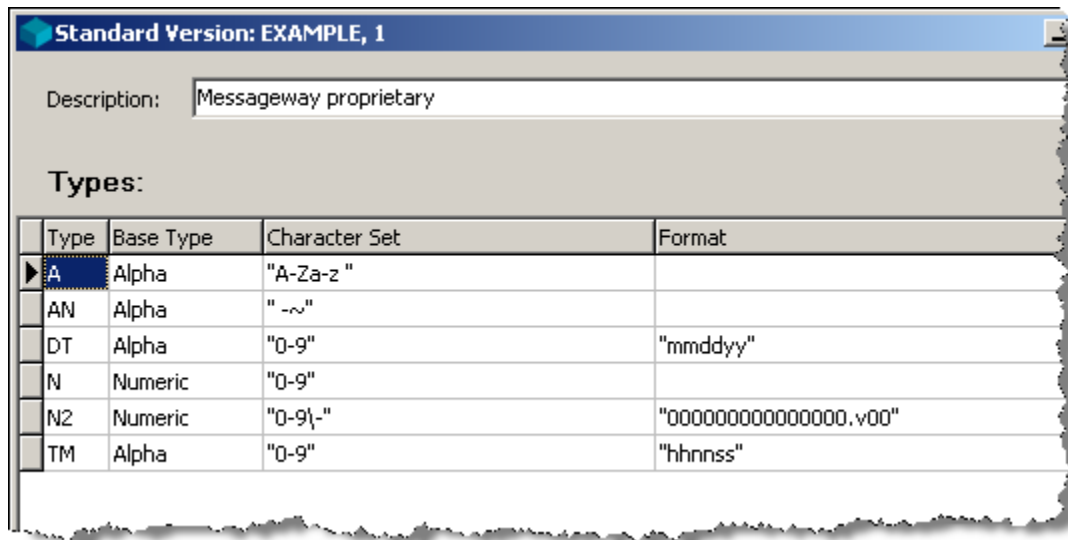
Task 1.1.2. Defining the Standard Types

Your next step is to *define the data types* (on page 646) for your standard.

IMPORTANT: The purpose of data types is to control the formatting of the output data. Data types defined here do not control how the input is treated. Input data is always treated as strings.

When you define your elements, you associate them with certain data types. You define these types based on 5 base types: Alpha, ID, DateTime, Numeric, and Binary. You also specify the set of valid characters for a type. Optionally, for date/time and numeric types, you can also define formats. These types are defined for a standard version and then associated with a document or wrapper when you generate the text file for the document or wrapper.

You enter standard types from the Standard Version window.



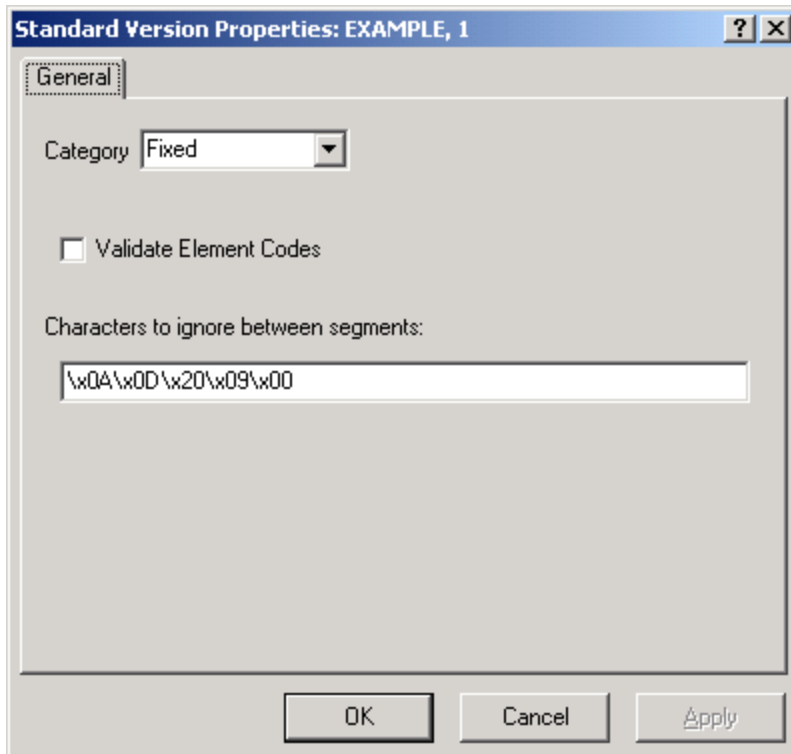
Task 1.1.3. Defining Standard Version Properties

You access Standard Version Properties from the Standard Version window by selecting the **Properties**



button from the toolbar. This is where you specify some processing parameters for this standard version.

- What parsing or generation techniques (Fixed, Delimited, SWIFT, XML) you are using
- Whether or not you want to validate elements
- Any special characters that appear between segments of fixed-length standard versions that you want the TRM to ignore during parsing.



Task 1.2. Defining the Content

The content of an EDI transmission is what is contained within the headers and trailers. The headers and trailers are the wrappers (envelope). Their contents are the documents. A document is the entity that contains the business information, such as a purchase order or an invoice.

Task Tree to Define the Content

Although you can define the content in any order, from the elements to the document or from the document to the elements, this discussion will show you the definitions from the document to the elements, so you can see the connections more easily.

Consider the following sub-tasks associated with this task. The boxes in white are optional. The striped boxes use defaults, but may require additional configuration.

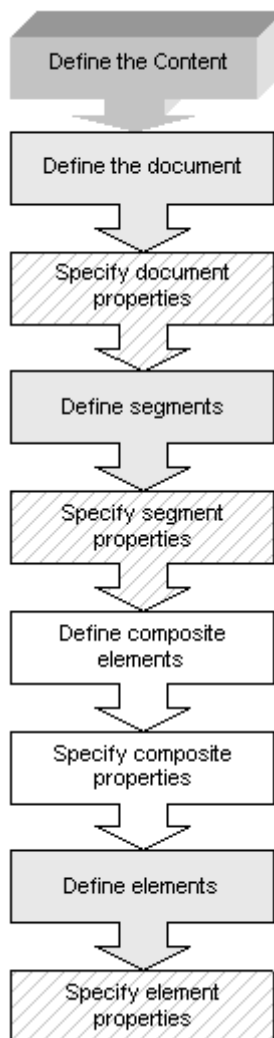


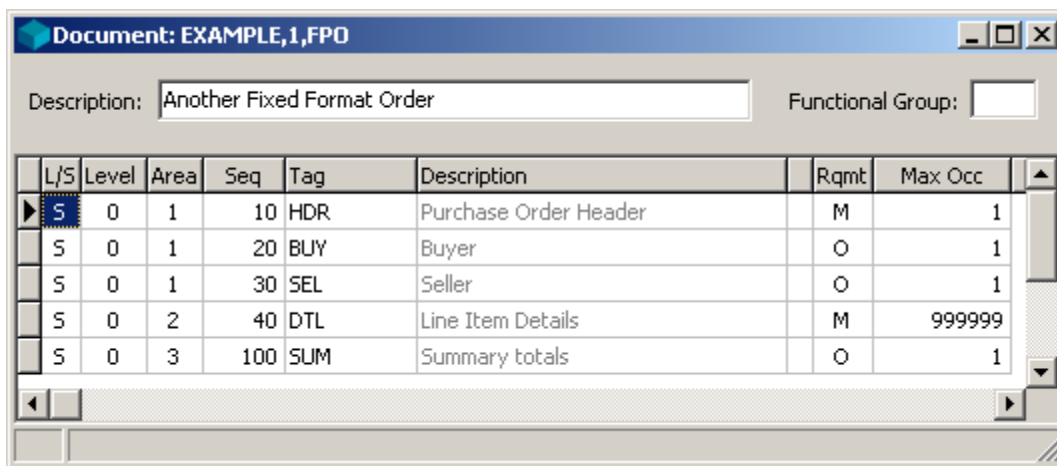
Figure 3. Sub-tasks to Define the Content (Defining Standards Task Tree)

Task 1.2.1. Defining the Document

Notice that the document definition contains the order of the segments that compose the documents. The dimmed information is actually entered from the Segment window.

IMPORTANT: Each document must contain at least one header segment.

Since you can enter Edibasic validation routines at several levels, such as the document, segment, and element, the blank column indicates when there are any such routines defined for this segment with a check mark.



Document: EXAMPLE,1,FPO

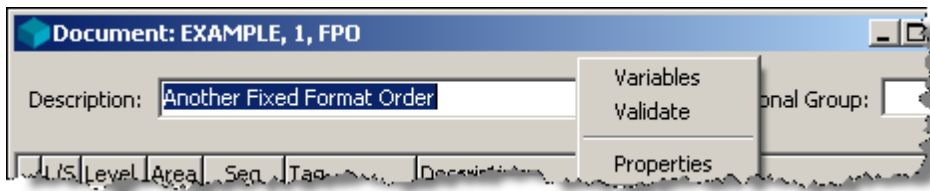
Description: Another Fixed Format Order Functional Group:

L/S	Level	Area	Seq	Tag	Description	Rqmt	Max Occ
5	0	1	10	HDR	Purchase Order Header	M	1
5	0	1	20	BUY	Buyer	O	1
5	0	1	30	SEL	Seller	O	1
5	0	2	40	DTL	Line Item Details	M	999999
5	0	3	100	SUM	Summary totals	O	1

You can write Edibasic validation routines at many different levels to be executed during parsing. To do so, you right click over the entity for which you want to write the routine, and then you select the **Validate** option from a pop-up menu. If you write a validation routine at the document level, by right clicking in the header area, this option will have a check mark next to it.

TIP: To view all validation routines, print a document report.

In the following figure, you can see that when you right click the header area of the document, the Validate option has no check mark, indicating that no validation routine has been written for this document.



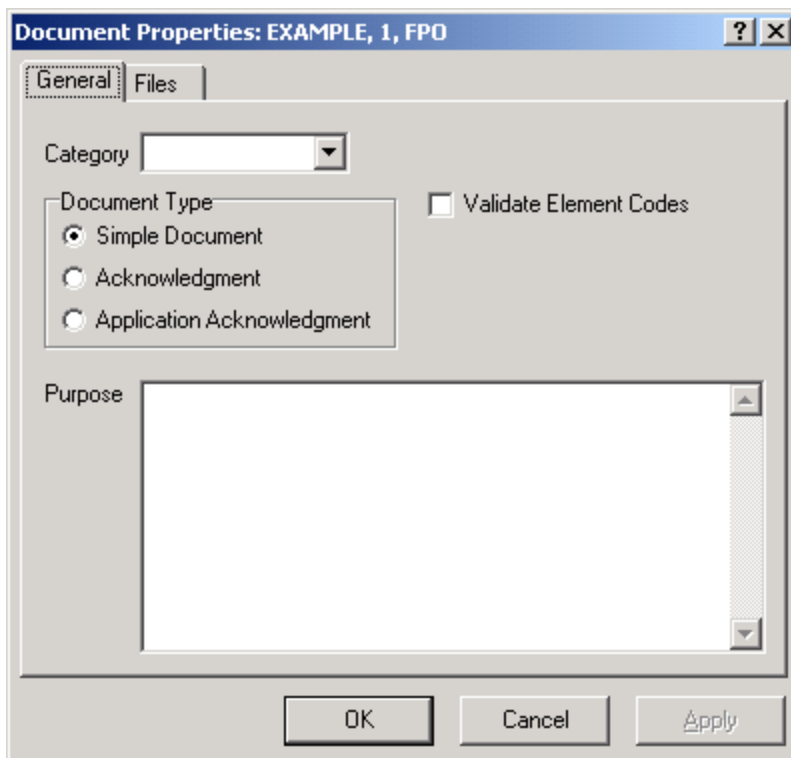
When a validation routine exists for the segment, a check mark appears in the blank column.

Task 1.2.2. Specifying Document Properties

The document properties are important. To access this window from the Document window, select the



Properties button from the toolbar. This is where you indicate the type of document, whether or not you want to validate element codes for this document, and if this is a document, acknowledgment, or application acknowledgment. If this document is an acknowledgment, there is another field you will see that allows you specify the level and an acknowledgment cross-reference file. Enter notes or other information for this document the **Purpose** box.

A screenshot of the 'Document Properties' dialog box. The title bar reads 'Document Properties: EXAMPLE, 1, FPO'. There are two tabs: 'General' (selected) and 'Files'. In the 'General' tab, there is a 'Category' dropdown menu. Below it is a 'Document Type' section with three radio buttons: 'Simple Document' (selected), 'Acknowledgment', and 'Application Acknowledgment'. To the right of this section is a checkbox labeled 'Validate Element Codes' which is unchecked. At the bottom of the dialog is a large text area labeled 'Purpose'. At the very bottom are three buttons: 'OK', 'Cancel', and 'Apply'.

Task 1.2.3. Defining the Segments

The next task is to define the segments that the document uses.

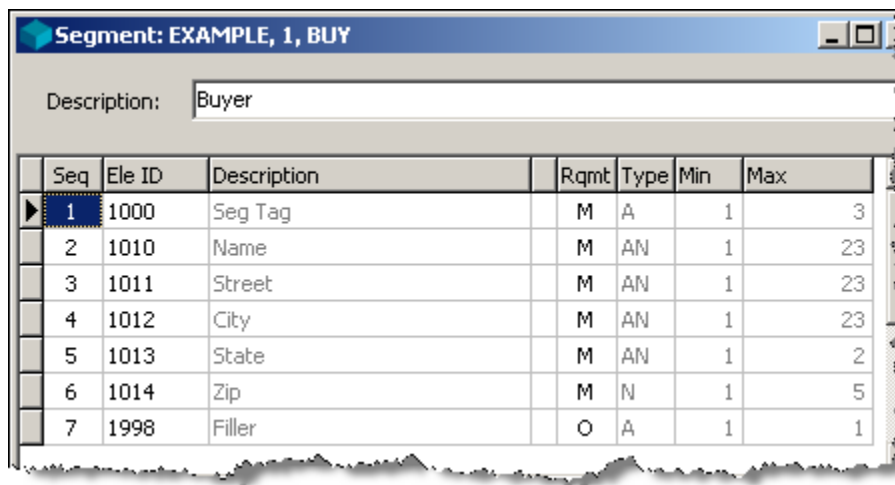
To create a new segment double-click any segment in the detail area of the Document window, or select the **Segments** folder in Data Explorer and then **Add** from the **File** menu.

When the New Segment dialog box appears, enter a value in the box.

IMPORTANT: The Segment Name must match the actual data to allow for proper identification during parsing. For example, if the value of the segment tag in the data is **HDR**, then the Segment name must be **HDR**.

The Segment window contains the elements that compose the segment and their order. Note that the shaded values are entered from the Element window.

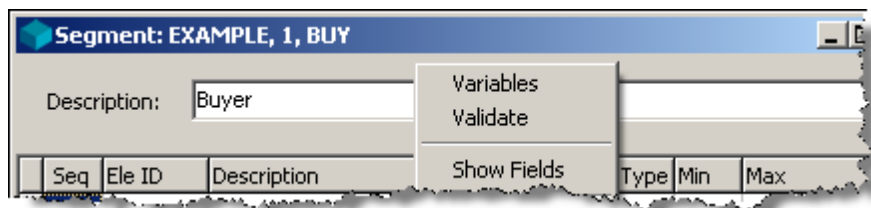
If an Edibasic user validation routine is entered for this segment, a check mark will appear on the pop-up menu when you right-click over the header area. If there is a validation routine entered for an element, there will be a check mark in the blank column.



Seq	Ele ID	Description	Rqmt	Type	Min	Max
1	1000	Seg Tag	M	A	1	3
2	1010	Name	M	AN	1	23
3	1011	Street	M	AN	1	23
4	1012	City	M	AN	1	23
5	1013	State	M	AN	1	2
6	1014	Zip	M	N	1	5
7	1998	Filler	O	A	1	1

You can write Edibasic validation routines at many different levels to be executed during parsing. To do so, you right-click over the entity for which you want to write the routine, and then you select the **Validate** option from the menu.

To determine whether a validation routine exists at the header level of a segment, right click in the header area. A pop-up menu appears, and when a routine exists, a check mark appears next to the **Validate** option.



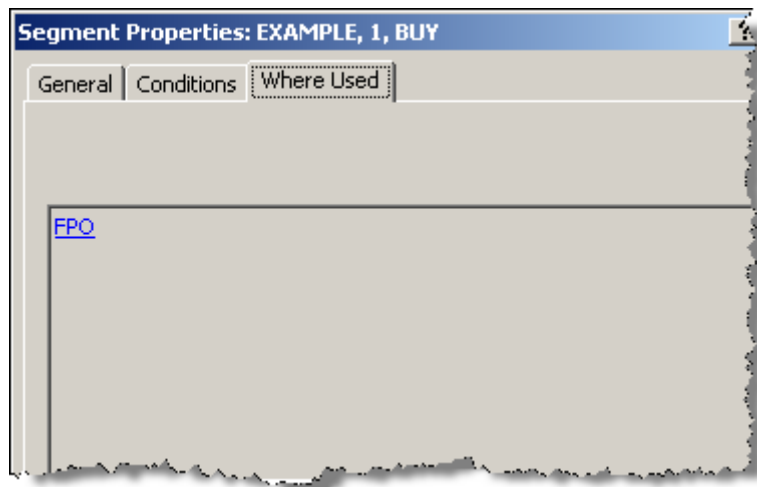
Seq	Ele ID	Description	Type	Min	Max

Task 1.2.4. Specifying the Segment Properties



To access the Segment Properties window from the Segment window, select the **Properties** button from the toolbar.

You can specify a different parsing or generation technique from that specified on higher entities on the **General** tab. You can specify conditions for this segment on the **Conditions** tab. You can determine which documents use this segment, by looking at the entries on the **Where Used** tab.

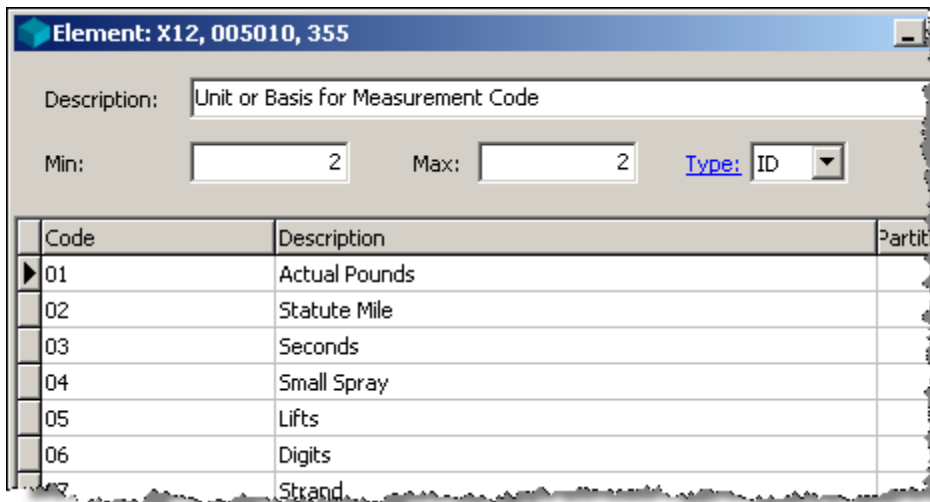


Task 1.2.5. Defining the Elements

The next task is to define the elements that each segment uses.

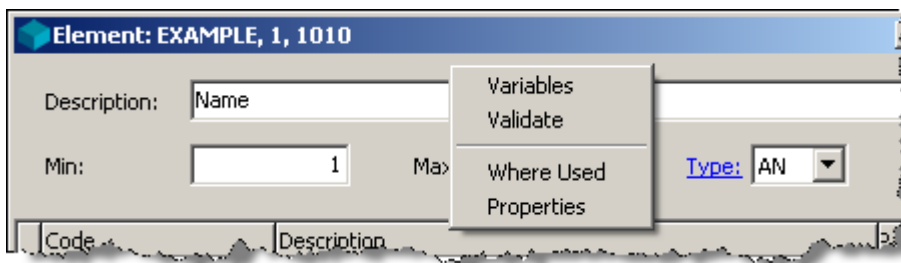
To create a new element, double-click any element in the detail area of the Segment window, or select the **Element** folder in Data Explorer and then **Add** from the **File** menu.

When you define elements, you can also specify a code list for an element. These code lists are also called element ID codes and are used most often by public standards, but seldom by proprietary standards. Codes typically specify the meaning of another entity, such as the date/time structure, or the type of reference number, such as purchase order or purchase order revision number, of a following or preceding element. Codes might also refer to an entire group of segments, or loop, such as whether the address is for a buyer or seller. The list is used to indicate what values are allowed for this element, which you can have the TRM validate during parsing. Codes can also refer to different parts of an element using partitions.



You can write Edibasic validation routines at many different levels to be executed during parsing. To do so, you right-click over the entity for which you want to write the routine, and then you select the **Validate** option from the menu.

To determine whether a validation routine exists at the header level of an element, right click in the header area. A pop-up menu appears, and when a routine exists, a check mark appears next to the **Validate** option.

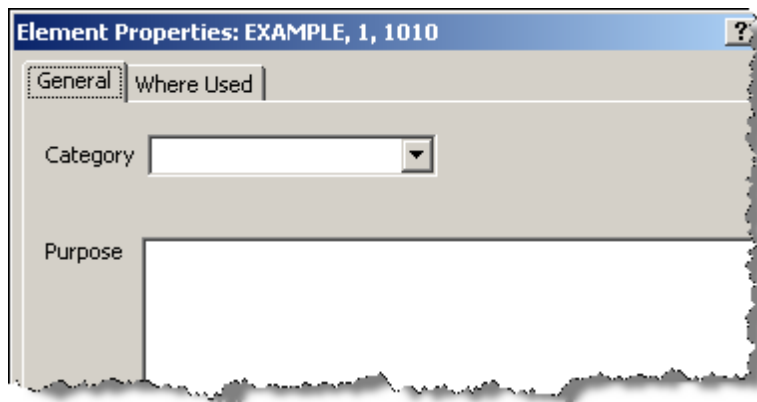


Task 1.2.6. Specifying Element Properties



To access the Element Properties window from the Element window, select the **Properties** button from the toolbar.

You can specify a different parsing or generation technique from that specified on higher entities on the **General** tab. You can determine which segments use this element, by looking at the entries on the **Where Used** tab.



Task 1.2.7. Generating the Text File for the Document

The TRM uses text files for processing. You can move the configuration information from a database structure to a text file from the Document window by selecting **Document** from the **Generate** menu. If the Document window is not currently open, you can also select your document in the right pane, and then select **Document** at the bottom of the **Generate** menu.

Task 1.3. Defining the Envelope (Wrappers)

The envelope of an EDI transmission comprises up to the 5 possible levels of segments that precede (header wrappers) and follow (trailer wrappers) the contents (documents). Their nested structure allows you to group types of contents. The outermost level is called the interchange level and the next level is called the functional group level. They may occur in pairs, as with the public standards, or there may be only header segments, as with the example proprietary standard. Their contents are the documents.

Task Tree to Define the Envelope

Although you can define the wrappers in any order, from the elements to the document or from the document to the elements, this discussion will again show you the definitions from the document to the elements.

NOTE: You do not have to define wrapper segments, but you must define a content header. An interchange without wrapper header or trailer segments is called a naked interchange or a null wrapper.

Consider the following sub-tasks associated with this task. The boxes in white are optional. The striped boxes use defaults, but may require additional configuration.

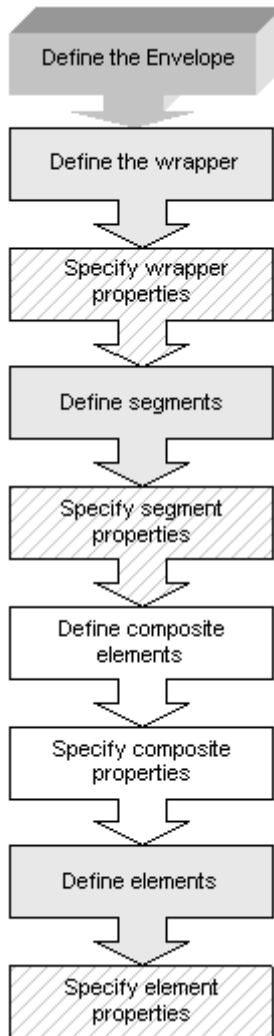


Figure 4. Sub-tasks to Define the Envelope (Defining Standards Task Tree)

Task 1.3.1. Defining the Wrapper

Notice that the wrapper definition contains the order of the segments that compose the wrapper, both headers and trailers, and there is a reference to the contents (document) header and trailer, although these content segments are actually defined as part of the document. The dimmed information is actually entered from the Segment window.

Since you can enter Edibasic validation routines at several levels, such as the wrapper, segment, and element, the blank column indicates if there are any such routines defined for this segment.

Wrapper: EXAMPLE,1,FWRAP

Description:

Header Segments:

Lvl	Seq	Seg Tag	Description	Rqmt
▶ 1	10	***	Fixed Header Record	M

Contents:

Header: [Purchase Order Header](#)

Trailer:

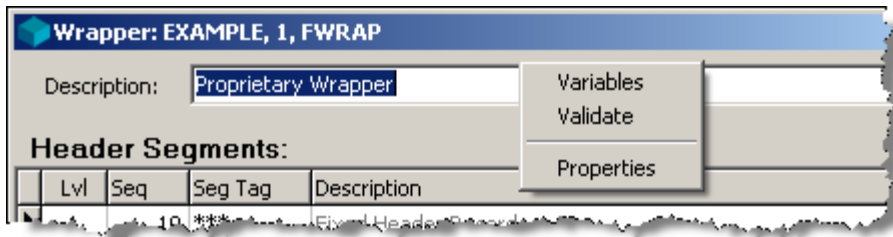
Trailer Segments:

Lvl	Seq	Seg Tag	Description	Rqmt
▶				

You can write Edibasic validation routines at many different levels to be executed during parsing. To do so, you right-click over the entity for which you want to write the routine, and then you select the **Validate** option from a pop-up menu. If you write a validation routine at the wrapper level, this option will have a check mark next to it. You can also see the validation routine when you print the wrapper report.

TIP: It is easier to view all validation routines by viewing a wrapper report.

In the following figure, you can see that when you right-click the header area of the wrapper, the **Validate** option has no check mark, indicating no validation routine has been written for this wrapper.



If there is a validation routine for the segment, a check mark appears in the blank column.

Task 1.3.2. Specifying Wrapper Properties

The wrapper properties are important. To access this window from the Wrapper window, select the

Properties button  from the toolbar.

From the **General** tab you specify:

- A parse/generate routine that is different from that specified for the standard version (**Category**)
- To validate element codes for this wrapper (**Validate Element Codes**)
- To parse input or generate output that has no breaks between segments (**Binary**) or that does have breaks between segments (**Text**) (**IO Mode**)
- To execute a user exit routine before it parses the data (**Pre-Process Method**)
- To execute a user exit routine after it generates the data (**Post-Process Method**)
- To specify a cross-reference file when creating acknowledgment wrappers (**Ack Xref**)

The screenshot shows a dialog box titled "Wrapper Properties: EXAMPLE, 1, FWRAP". It has four tabs: "General", "Service Characters", "StdID Match", and "Files". The "General" tab is selected. The dialog contains the following fields and controls:

- Category:** A dropdown menu.
- Validate Element Codes:** A checkbox that is currently unchecked.
- IO Mode:** A group box containing two radio buttons: "Binary" (which is selected) and "Text".
- Pre-Process Method:** A text input field.
- Post-Process Method:** A text input field.
- Ack XRef:** A text input field.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

From the **Service Characters** tab if this is a delimited standard, you specify:

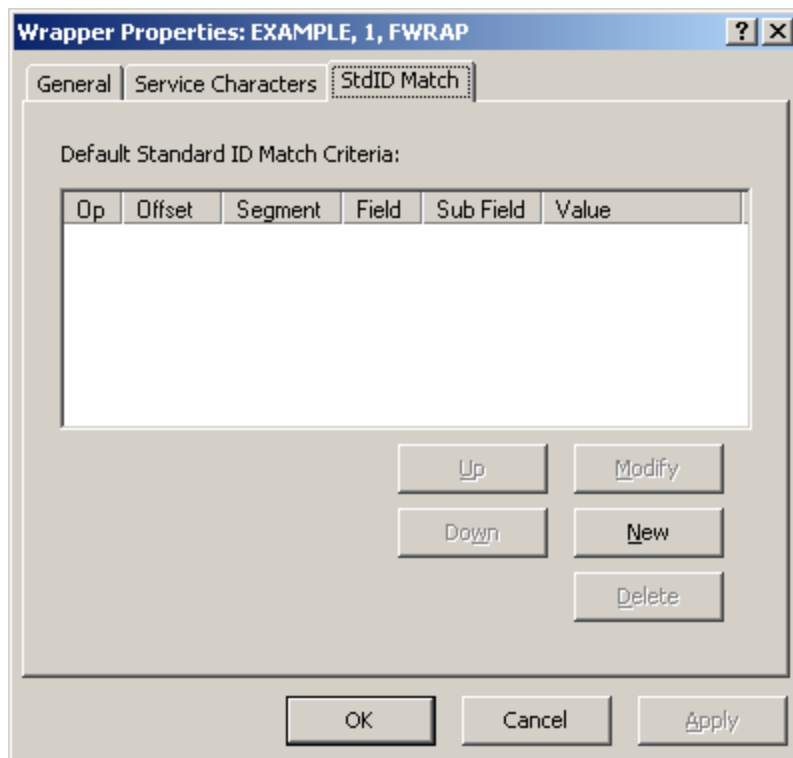
- To use the offset to find the service characters to use for parsing
- Failing an offset, to use the values provided for parsing
- Use the values provided for generation, which can be overridden from the Trade Agreement Properties configuration

The screenshot shows a dialog box titled "Wrapper Properties: EXAMPLE, 1, FWRAP". It has three tabs: "General", "Service Characters", and "StdID Match". The "Service Characters" tab is selected. The dialog contains a table with two columns: "Value" and "Offset". The rows are labeled with various service characters.

	Value	Offset
Segment Terminator	~	
Tag Delimiter	=	
Element Delimiter	*	
Component Delimiter	:	
Repetition Separator		
Release Character	?	
Decimal Mark		

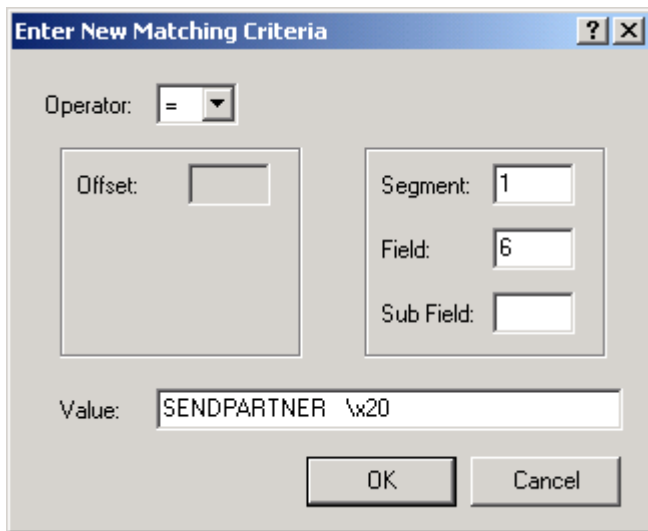
At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

From the **StdID Match** tab, you can specify which default criteria for the standard identification process.



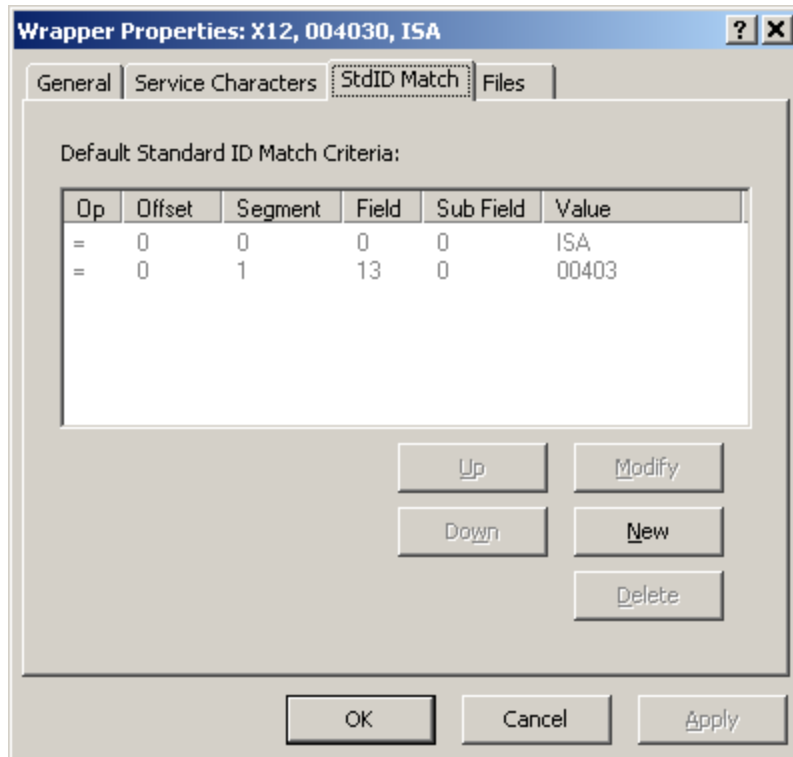
When you select **New**, the **Enter New Matching Criteria** dialog box appears. You may then enter the new matching criteria.

IMPORTANT: Trailing spaces entered for the value are deleted. To maintain trailing spaces to match an input value with trailing spaces, you may use a hex value at the end of the value field. For example, for an element of 15 characters, where the input value is **SENDPARTNER** followed by 4 spaces, in the Value field of the **Enter New Matching Criteria** dialog box you would enter **SENDPARTNER \x20**, which is **SENDPARTNER** followed by three spaces terminated with a hex space.

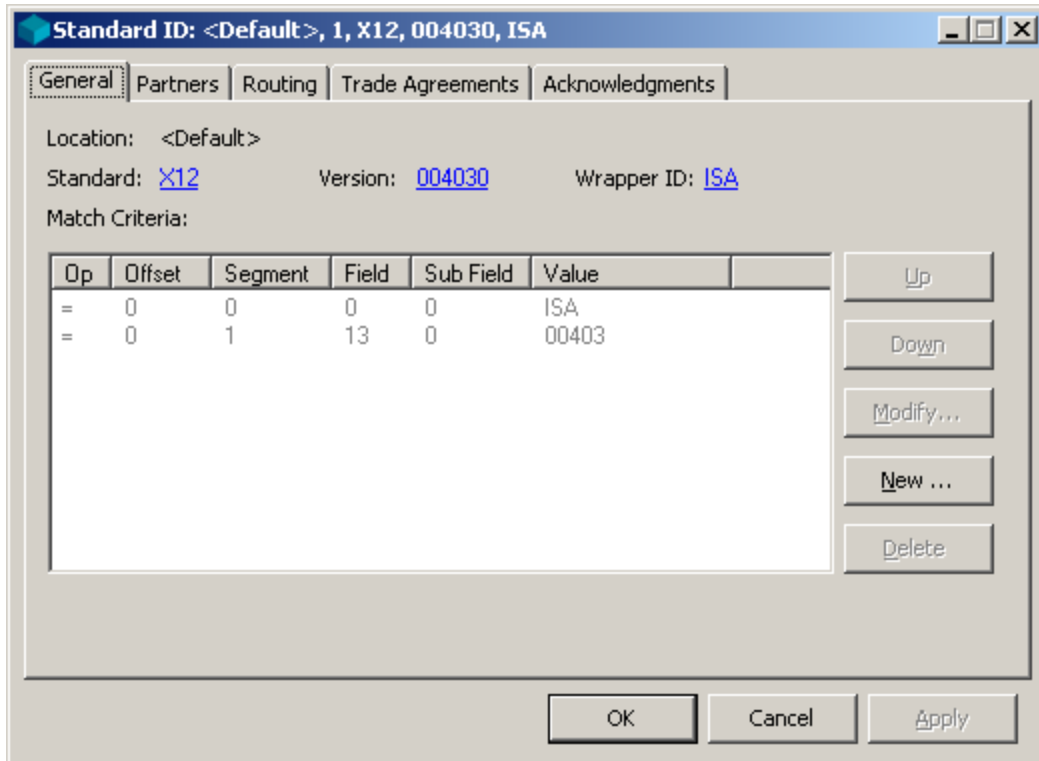


When you add a new wrapper to a location in Partner Explorer for the standard identification process, these default values are automatically entered as matching criteria on the **General** tab of the Standard ID window.

For example, look at the **StdID Match** tab of the Wrapper Properties window for the ISA wrapper, X12 004030 standard version. Although the standard is delivered with these default values entered, you could enter default values for your proprietary standard by using the buttons on the window.



Then when you add a wrapper to a location, the default criteria appear as matching criteria on the **General** tab, as you can see here.



Task 1.3.3. Defining the Segments

The next task is to define the segments that the wrapper uses.

To create a new segment you can double-click any segment in the detail area of the Wrapper window, or you can select the Segment folder in Data Explorer and then **Add** from the **File** menu.

The Segment window contains the elements that compose the segment and their order. Note that the dimmed values are entered from the Element window.

The **Field** column automatically appears so you can store the incoming parsed data in internal fields that the TRM can then use to find partnership information and an appropriate trade agreement profile.

When an Edibasic user validation routine is entered for this segment, a check mark will appear on the menu when you right-click over the header area. When there is a validation routine entered for an element, there will be a check mark in the blank column.

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	0008	Record Tag (3 character)		M	AN	1	3
2	0002	Partner	Rec Partner ID	M	AN	1	12
3	0201	Transaction Set ID	Document ID	M	AN	1	3

For delimited segments, the segment tag is always at the beginning of the segment, and it is never defined as an element. There is no other choice of location for the segment tag of a delimited segment.

For fixed format segments, the segment tag is by default the first element of the segment. There may be times when users want to define an element other than the first element as the segment tag. To do this, users must explicitly define the segment as a fixed format segment.

For example, assume we want to define the second element in the BUY segment as the tag. The logic of this choice is questionable, because you would want to choose a tag whose value would be constant. We are using this for purpose of discussion only. First we select the HDR segment, and then select **Fixed** as the category. Doing this displays the **Tag** column on the Segment window.

The following window shows the BUY segment, which is a fixed format segment. The category is set at a higher level, for the entire standard version. The segment itself is not explicitly defined as fixed, so the **Tag** column does not appear on the window.

Seq	Ele ID	Description	Rqmt	Type	Min	Max
1	1000	Seg Tag	M	A	1	3
2	1010	Name	M	AN	1	23
3	1011	Street	M	AN	1	23
4	1012	City	M	AN	1	23
5	1013	State	M	AN	1	2
6	1014	Zip	M	N	1	5
7	1998	Filler	O	A	1	1

Now we will explicitly set the category to **Fixed** for the segment on the Segment Properties window.



Notice that the **Tag** column appears.

The screenshot shows the 'Segment: EXAMPLE, 1, BUY' window. The 'Description' field contains 'Buyer'. Below it is a table with the following data:

Seq	Ele ID	Description	Tag	Rqmt	Type	Min	Max
1	1000	Seg Tag		M	A	1	3
2	1010	Name		M	AN	1	23
3	1011	Street		M	AN	1	23
4	1012	City		M	AN	1	23
5	1013	State		M	AN	1	2
6	1014	Zip		M	N	1	5
7	1998	Filler		O	A	1	1

To set the second element, the Purchase Order Number, as the segment tag, you enter any value in the tag field for that element. This field is a toggle field. When an element is selected as the tag for the segment, a **T** appears in this column. The value in this element is now the segment tag for the element.

Seq	Ele ID	Description	Tag	Rqmt	Type	Min	Max
1	1000	Seg Tag		M	A	1	
2	1010	Name	T	M	AN	1	23
3	1011	Street		M	AN	1	23
4	1012	City		M	AN	1	23
5	1013	State		M	AN	1	2
6	1014	Zip		M	N	1	5
7	1998	Filler		O	A	1	1

To deselect a tagged element, place your cursor in the field and enter any value. The **T** will disappear.

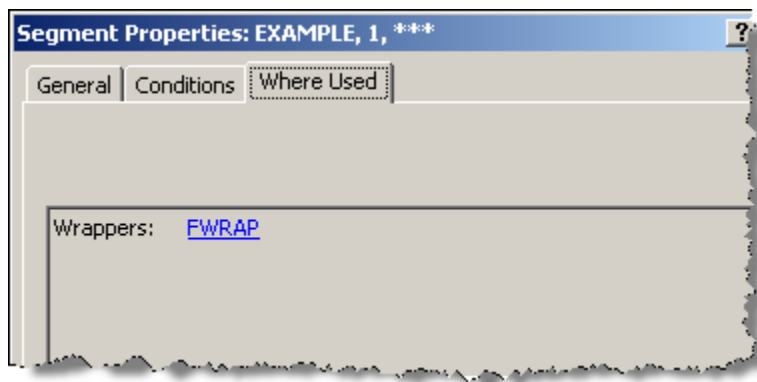
Task 1.3.4. Specifying the Segment Properties

You access the Segment Properties window from the Segment window by selecting the **Properties** button



from the toolbar.

You can specify a different parsing or generation technique from that specified on higher entities on the **General** tab. You can specify conditions for this segment on the **Conditions** tab. You can determine which wrappers use this segment, by looking at the entries on the **Where Used** tab.



Task 1.3.5. Defining the Elements

The next task is to define the elements that each segment uses.

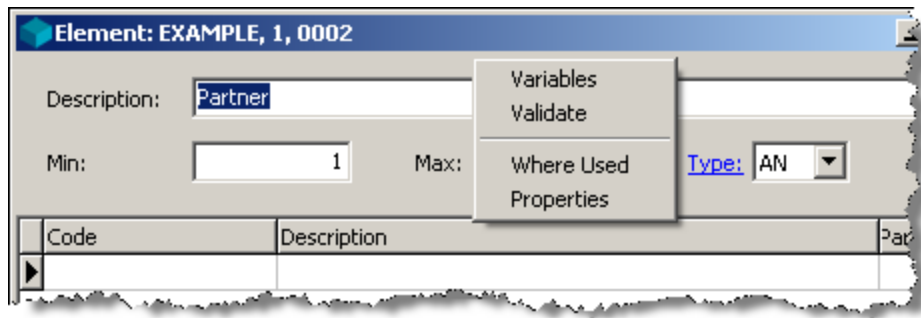
To create a new element you can double-click any element in the detail area of the Segment window, or you can select the **Element** folder in Data Explorer and then **Add** from the **File** menu.

When you define elements, you can also specify a list of codes, also called element ID codes, for an element. These code lists are used most often by public standards. Codes typically specify the meaning of another entity, such as the date/time structure, or the type of reference number, such as purchase order or purchase order revision number, of a following or preceding element. Codes might also refer to an entire group of segments, or loop, such as whether the address is for a buyer or seller. The list is used to indicate what values are allowed for this element, which can be used to validate incoming data during parsing. Codes can also refer to specific parts of an element by assigning them to partitions.

Code	Description	Partit
01	Actual Pounds	
02	Statute Mile	
03	Seconds	
04	Small Spray	
05	Lifts	
06	Digits	
07	Strand	

You can write Edibasic validation routines at many different levels to be executed during parsing. To do so, you right-click over the entity for which you want to write the routine, and then you select the **Validate** option from the menu.

If you write a validation routine at the element level, by right clicking in the header area, this option will have a check mark next to it. Validate routines written for an element window are limited to the element header area.

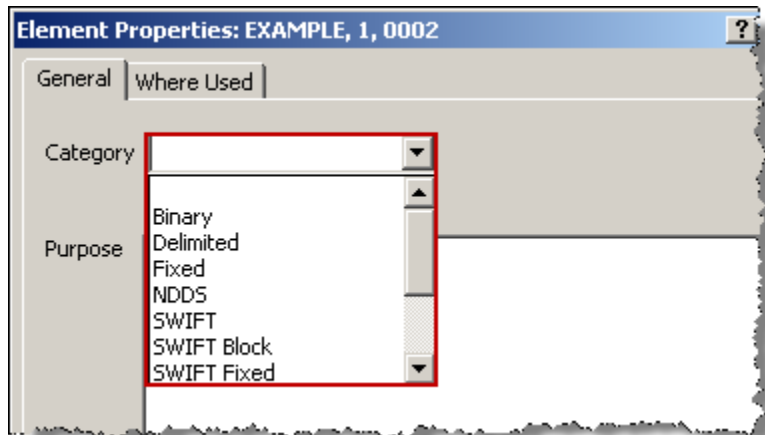


Task 1.3.6. Specifying Element Properties

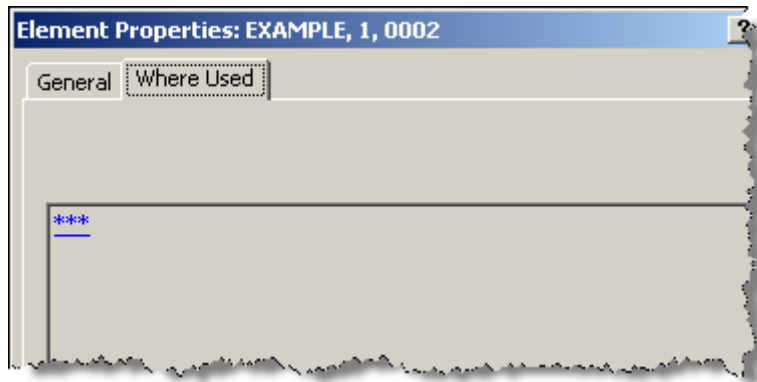


To access the Element Properties window from the Element window, select the **Properties** button from the toolbar.

You can specify a different parsing or generation technique from that specified on higher level entities on the **General** tab.



You can determine which segments use this element, by looking at the entries on the **Where Used** tab.






Task 1.3.7. Generating the Text File for the Wrapper

The TRM uses text files for processing. You can move the configuration information from a database structure to a text file from the Wrapper window by selecting **Generate>Document...** from the menu bar. If your Wrapper window is not currently open you can also select your wrapper in the right pane, and then select **Document** at the bottom of the **Generate** menu.

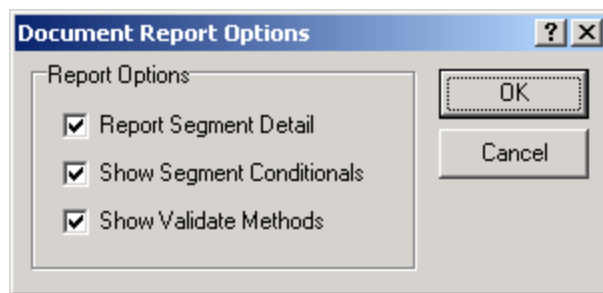
Your next major task will be to create the maps for your wrapper and document, which is only required if you are translating. If you are not doing translation, you can skip this task. You can then define a trade agreement profile, and, if required, an acknowledgment profile.

Task 1.4. Printing Document and Wrapper Reports

You can review your definitions by printing a report. To print a report, select your document or wrapper in the right pane of Data Explorer. From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file
- The **Print Preview** button  to print reports to the screen
- The **Print to PDF** button  to print to a PDF file

From the dialog box you select the degree of detail you want on your report.



The first page shows a list of segments for the document or wrapper in order of definition.

MW Translator Workbench Document Report						
EXAMPLE, 1 <Fixed>						
FPO, Another Fixed Format Order						
(Simple Document)						
Area	Seq	Tag	Description	Reqm't	Max	
1	10	HDR	Purchase Order Header	M	1	
1	20	BUY	Buyer	O	1	
1	30	SEL	Seller	O	1	
2	40	DTL	Line Item Details	M	999999	
3	100	SUM	Summary totals	O	1	

Document Definition Filenames

Workbench Filename: appl/EXFPO1.txt
Target Filename: APP.EXFPO1

Another page will show you the definitions of the elements, if you have selected **Show Segment Details**. The segments appear in sorted order. The following figure is a partial report of the elements.

MW Translator Workbench Document Report							
EXAMPLE, 1							
FPO, Another Fixed Format Order							
BUY		Buyer					
Seq	Ele ID	Description	Reqm't	Type	Min	Max	
1	1000	Seg Tag	M	A	1	3	
2	1010	Name	M	AN	1	23	
3	1011	Street	M	AN	1	23	
4	1012	City	M	AN	1	23	
5	1013	State	M	AN	1	2	
6	1014	Zip	M	N	1	5	
7	1998	Filler	O	A	1	1	
DTL		Line Item Details					
Seq	Ele ID	Description	Reqm't	Type	Min	Max	
1	1000	Seg Tag	M	A	1	3	
2	1020	Line Number	M	N	1	6	
3	1021	Description	M	AN	1	24	
4	1022	Quantity	M	N	1	6	
5	1023	Unit Cost	M	N2	1	12	
6	1024	Filler	O	A	1	29	

Special Cases. Defining Standards

The following discussions provide configuration examples to meet special requirements.

Configuring Null Wrappers

Typically, you configure wrappers to specify partner information and allow for multiple levels containing multiple instances of documents, which are features required by the public standards. For proprietary standards, you may not need partner information to identify the inbound wrapper, and you may not use multiple levels of wrapper information. Although you must always define a wrapper, you can define a null wrapper. A null wrapper contains no interchange or functional group-level segments. However, you must adhere to the following requirements:

- Define a wrapper that contains a contents header, which can be a dummy segment that is not even part of the input document.
- Not use inbound partner IDs to find a trade agreement profile, because the trade agreement profile will be associated with the inbound location, using the standard ID record.

The inbound null wrapper has additional configurations that you must supply in order for it to work. Here are the steps you must follow:

- 1 *Define a null wrapper* (on page 146).
- 2 *Define a trade agreement that will be used to process your document* (on page 148).
- 3 *Associate the trade agreement with a standard ID record* (on page 149).

To Define a Null Wrapper

You may or may not need the same wrapper definition when the document is inbound rather than outbound. Assuming you already have a standard defined with standard types, follow these steps:

- 1 Create a wrapper that has no wrapper header or trailer segments, but specify at least a content header segment, which can be an existing segment unrelated to the document.
- 2 If you have a content header segment to use, enter that segment tag in the **Contents Header** box.
– otherwise –

If you follow this example, you will also have to create a dummy segment with a dummy element.

The image shows two screenshots of a configuration tool. The top window is titled "Wrapper: MYSTD, Test1, NULLIN". It has a "Description:" field containing "Null Wrapper Inbound". Below this is a section for "Header Segments:" with a table that is currently empty. The table has columns for "Lvl", "Seq", "Seg Tag", "Description", and "Rqmt". Below the header segments is a "Contents:" section with "Header:" set to "DUM" (with a blue link "Dummy Segment" next to it) and "Trailer:" set to an empty field. Below the contents is a section for "Trailer Segments:" with another empty table with the same columns as the header segments table. The bottom window is titled "Segment: MYSTD, Test1, DUM". It has a "Description:" field containing "Dummy Segment". Below this is a table with columns: "Seq", "Ele ID", "Description", "Rqmt", "Type", "Min", and "Max". The table contains one row with the following values: "*" in the "Seq" column, "1" in "Ele ID", "DUM" in "Description", "Dummy Element" in "Description", an empty cell in "Rqmt", "AN" in "Type", "1" in "Min", and "3" in "Max".

Wrapper: MYSTD, Test1, NULLIN

Description: Null Wrapper Inbound

Header Segments:

Lvl	Seq	Seg Tag	Description	Rqmt
-----	-----	---------	-------------	------

Contents:

Header: DUM [Dummy Segment](#)

Trailer:

Trailer Segments:

Lvl	Seq	Seg Tag	Description	Rqmt
-----	-----	---------	-------------	------

Segment: MYSTD, Test1, DUM

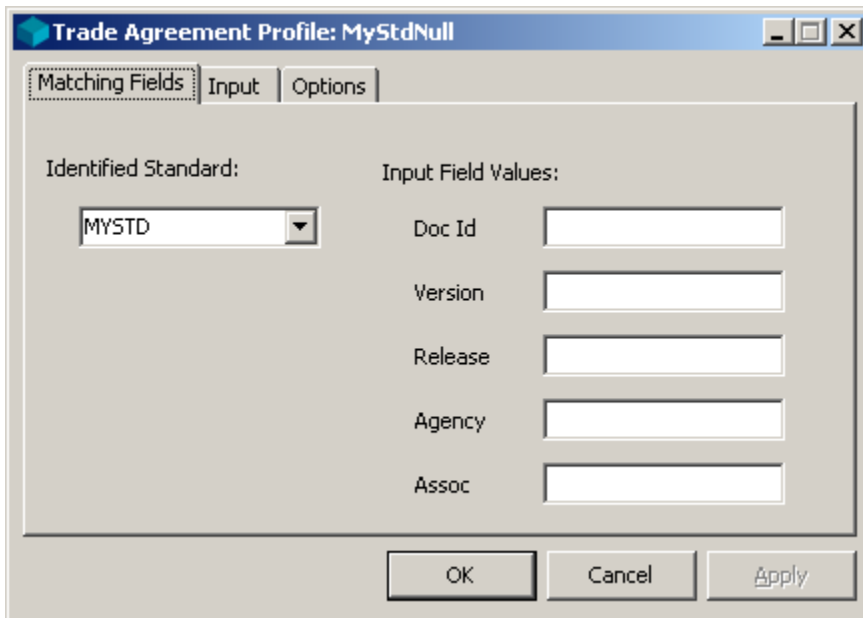
Description: Dummy Segment

Seq	Ele ID	Description	Rqmt	Type	Min	Max	
*	1	DUM	Dummy Element		AN	1	3

- 3 Close the window to save your configuration.

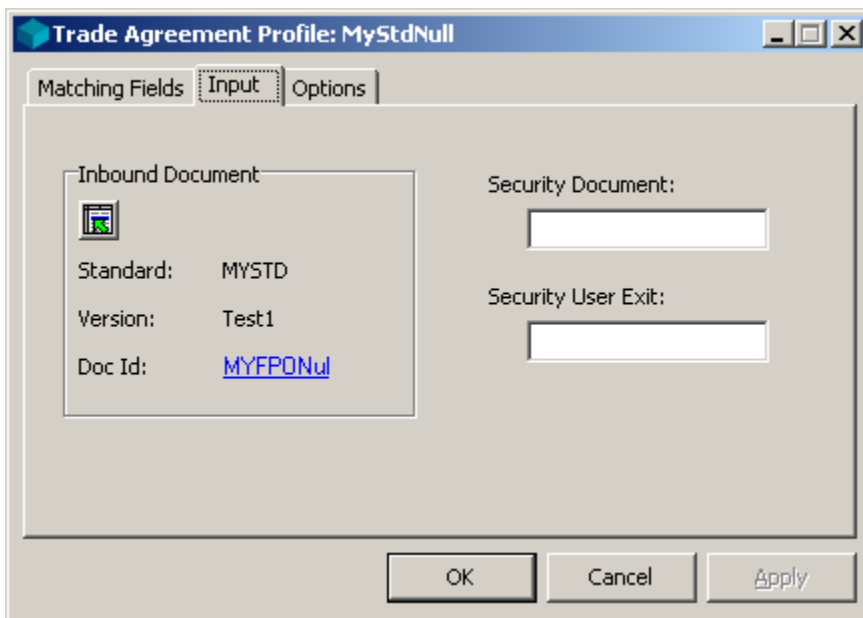
To Specify a Trade Agreement for the Inbound Null Wrapper

- 1 Create a trade agreement that specifies only the standard. The input field values must be blank, because there are no values being stored in the associated internal fields with which they must match. Specifically, the **Document ID** internal field will be null, because there is no value to be stored from an incoming wrapper, including the content header segment, such as **DUM**.



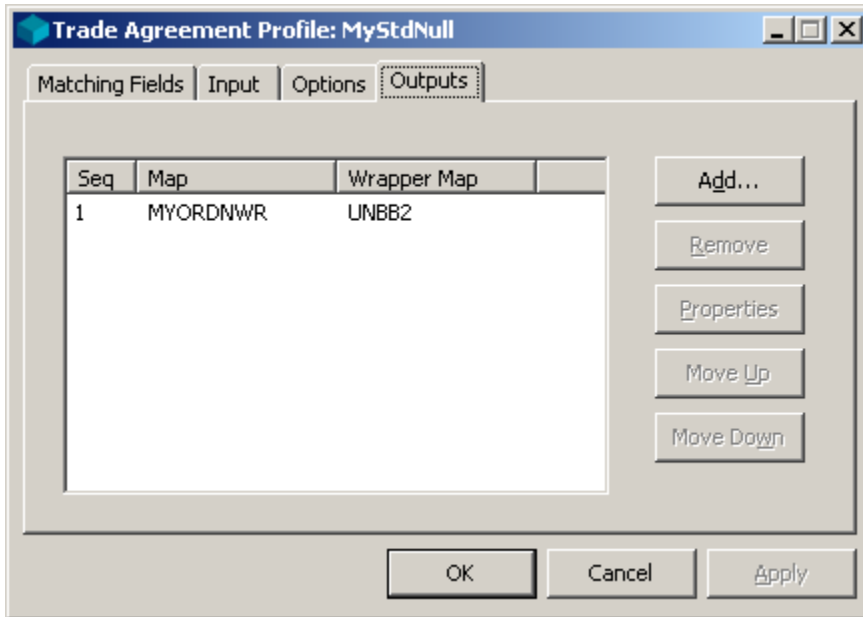
The screenshot shows the 'Trade Agreement Profile: MyStdNull' dialog box with the 'Matching Fields' tab selected. The 'Identified Standard' dropdown is set to 'MYSTD'. The 'Input Field Values' section contains five empty text boxes for 'Doc Id', 'Version', 'Release', 'Agency', and 'Assoc'. The 'OK', 'Cancel', and 'Apply' buttons are visible at the bottom.

- 2 On the **Input** tab, specify the inbound document definition.



The screenshot shows the 'Trade Agreement Profile: MyStdNull' dialog box with the 'Input' tab selected. The 'Inbound Document' section displays a document icon and the following details: Standard: MYSTD, Version: Test1, and Doc Id: MYFPDNull. The 'Security Document' and 'Security User Exit' sections each have an empty text box. The 'OK', 'Cancel', and 'Apply' buttons are visible at the bottom.

- 3 Enter whatever you want on the **Options** and **Outputs** tabs. For more information, refer to the topic, *Task 3. Defining Trade Agreement Profiles* (on page 222).



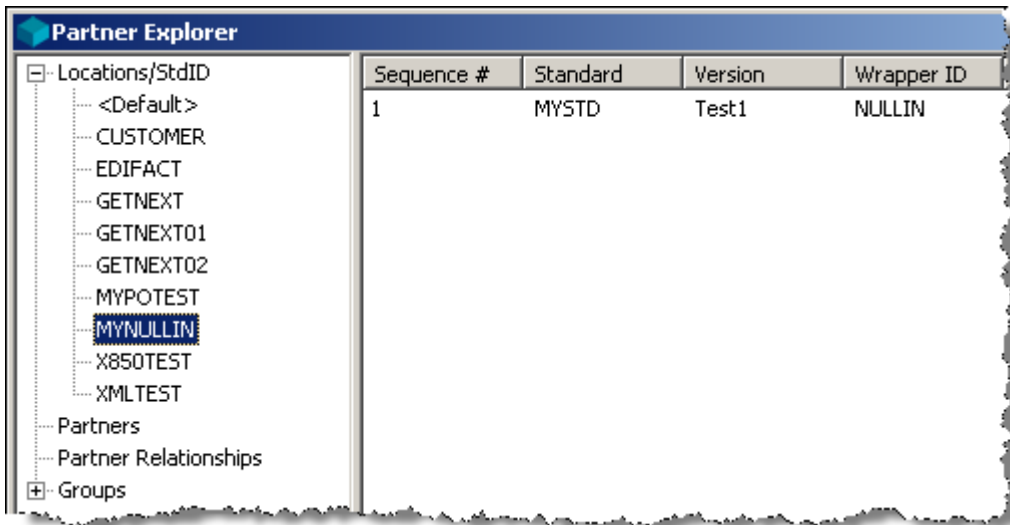
- 4 Select **OK** to close the window to save the configuration.

To Specify a Standard ID to Identify the Inbound Null Wrapper

You must configure the location and wrapper so that the trade agreement can be linked to the standard ID, since you have no partner information that would allow you to link the trade agreement to a partner definition.

- 1 Create a special location and define the null wrapper in that location.
– or –
Define the wrapper in the **<Default>** location.

NOTE: You must be sure that it is correctly defined with matching criteria and that it is in the correct location in the list that it will be selected within the <Default> location. To avoid selection problems, you may create a special location and define your null wrapper in that location. You must make sure that MessageWay passes this location name a location for MessageWay. The location name must also conform to the MessageWay requirements, such as 8 characters maximum, uppercase, for MessageWay when it runs on a UNIX or Linux platform. If you cannot pass a special location name to the MessageWay adapter you use, you must define the wrapper in the <Default> location.



The screenshot shows the 'Partner Explorer' window. On the left is a tree view under 'Locations/StdID' with the following items: <Default>, CUSTOMER, EDIFACT, GETNEXT, GETNEXT01, GETNEXT02, MYPOTEST, MYNULLIN (highlighted), X850TEST, XMLTEST, Partners, Partner Relationships, and Groups. On the right is a table with the following data:

Sequence #	Standard	Version	Wrapper ID
1	MYSTD	Test1	NULLIN

- 2 You can leave the **General** tab blank, if this is the only wrapper in the location.

The screenshot shows a dialog box titled "Standard ID: MYNULLIN, 1, MYSTD, Test1, NULLIN". The "General" tab is selected, showing the following information:

- Location: MYNULLIN
- Standard: [MYSTD](#)
- Version: [Test1](#)
- Wrapper ID: [NULLIN](#)

Match Criteria:

Op	Offset	Segment	Field	Sub Field	Value	

Buttons on the right side of the table:

- Up
- Down
- Modify...
- New ...
- Delete

Buttons at the bottom of the dialog box:

- OK
- Cancel
- Apply

- 3 On the **Partners** tab, you must leave the **Partner Definition Required** boxes unchecked, because the internal field values for partners will be null after parsing.

Since they will be null, you can specify default partner definitions, if you wish. You could then associate the trade agreement with the default recipient partner, rather than the standard ID, as we do in the following step. Otherwise, leave everything blank.

Standard ID: MYNULLIN, 1, MYSTD, Test1, NULLIN

General Partners Routing Trade Agreements Acknowledgments

Location: MYNULLIN
Standard: [MYSTD](#) Version: [Test1](#) Wrapper ID: [NULLIN](#)

Partner Definition Required

Sending Partner Recipient Partner

Default Sending Partner

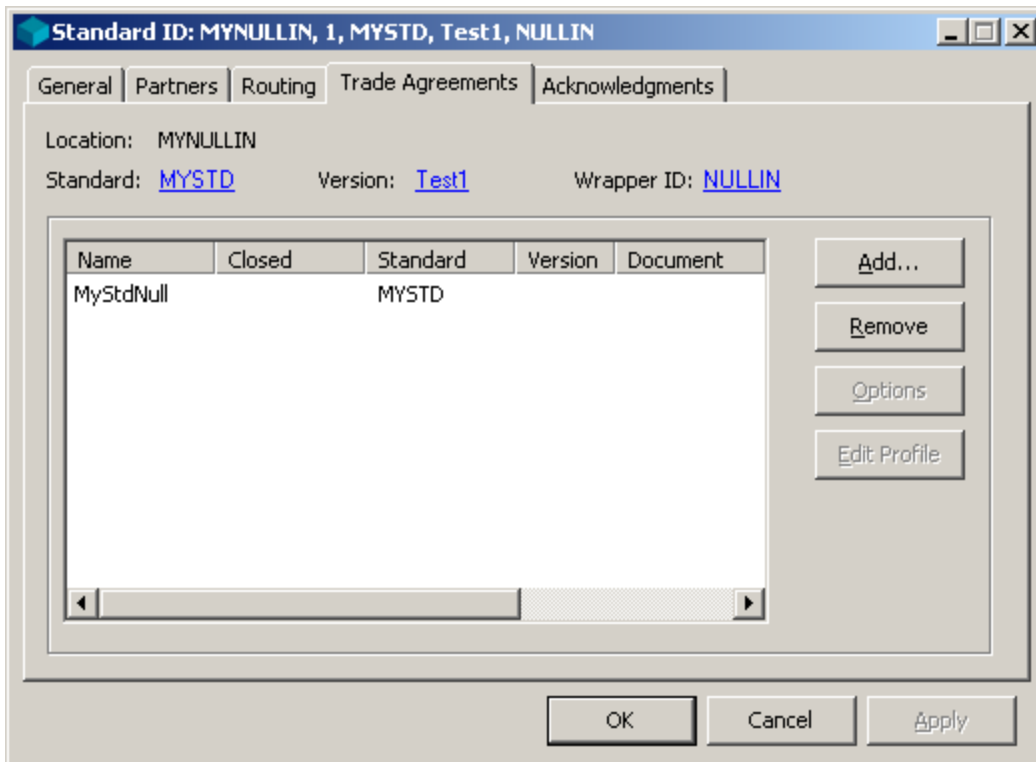
Name: Partner2-EDIFACT

Default Recipient Partner

Name: Partner1-EDIFACT

OK Cancel Apply

- 4 On the **Trade Agreements** tab, you select the trade agreement for your null wrapper.



- 5 Close the window to save your configurations.

Configuring Variable-length Segments in Fixed-length Standards

Some fixed-length formats actually have variable-length segments. Variable-length segments may result from truncation, because:

- Optional elements at the end of a segment have no data
 - or –
- A final element in a segment may contain fewer significant characters than the maximum allowed, that is, it is a variable-length element

For these definitions to work, the IO Mode on the Wrapper Properties window must be **Text**, not **Binary**. The IO Mode controls parsing and generation processes. With **Binary** mode, MW Translator will use the sum of the length of the elements to determine the end of the segment. With **Text** mode, MW Translator uses an end-of-segment marker, such as linefeed, carriage-return/linefeed, or carriage return, which it converts internally to a linefeed character. In binary mode, when MW Translator reads short segments it will not parse the data correctly, because it will read past the actual end of the shortened segment into the next segment of data. Therefore, to correctly parse short segments, you must have the **IO Mode** set to **Text** and use one of the segment termination markers mentioned earlier.

To Allow Missing Trailing Elements

First, make sure the IO mode on the Wrapper Properties window is set to **Text** and that the data contains an end-of-segment marker: Linefeed, carriage-return, or carriage-return/linefeed.

- 1 Within Data Explorer, expand the **Standards** folders, and select **Segments** within the appropriate standard and version.
- 2 In the right pane, double-click the segment.
The Segment window appears.
- 3 For the last element of the segment, set the requirement (**Rqmt** column) to optional (**O**).
When you do not map data to the last element in a segment, MW Translator does not generate the element, which produces a short segment.
- 4 Close all windows, and regenerate your definitions from the **Generate** menu.

To Allow Short Trailing Elements

First, make sure the IO mode on the Wrapper Properties window is set to **Text** and that the data contains an end-of-segment marker: Linefeed, carriage-return, or carriage-return/linefeed.

- 1 Within Data Explorer, expand the **Standards** folders, and select **Segments** within the appropriate standard and version.
- 2 In the right pane, double-click the segment.
The Segment window appears.
- 3 For the last element of the segment, set the requirement (**Rqmt** column) to optional (**O**).
- 4 For output, to allow MW Translator to generate short segments:
 - a) Double-click the last element.
The Element window appears.



- b) Select the **Properties** button from the toolbar.
The Element Properties window appears.
 - c) On the **General** tab, for **Category**, scroll down and select **Variable** from the list.
The **Variable** category allows MW Translator to truncate the element during generation.
For information about truncation rules for data types defined in Standard Version window, refer to the topic, *Base Type (Types)* (on page 643).
For information about the various categories available for elements defined in the Element Properties window, refer to the topic, *Category* (on page 596).
- 5 Close all windows, and regenerate your definitions from the **Generate** menu.

Creating Maps

Overview



If you are only doing routing or validation and routing, you do not need to create maps. If you are translating, you do. Mapping is the process of identifying all of the information you need to create your output document. Most of the information will come from the input document directly. Some will be a result of calculations. Some will be literals that you supply, because they are not available from the input.





Types of Mapping

There are two types of mapping you can use to create your output: visual and Edibasic. Visual mapping is a combination of techniques: drag-and-drop, map once, map from a literal, map from internal fields. These techniques allow you to visually relate input loops, segments, and elements to output loops, segments, and elements.

Visual Mapping

The following table lists the tasks you can perform using visual techniques, the effect each has, and explains what source statement is associated with it. The source statement is visible in the status area, located at the bottom of the Map window.

Visual Mapping Task	Effect	Icon	Source Statement
Create a loop or a repeating or non-repeating segment from a repeating input segment or loop.	generates same number of items as in the input.		Repeats from
Create a non-repeating segment or loop from a repeating or non-repeating segment or loop.	generates no or one occurrence depending on the existence of the input (uses first occurrence).		Depends on

Visual Mapping Task	Effect	Icon	Source Statement
Create only one occurrence of the segment or loop.	generates one occurrence of loop or segment, regardless of number of occurrences of input.		Once
Create element from a literal value, from an internal field, or from any input element or other information in the data element store.	For numeric and date/time types, elements are mapped with data conversion (if formats are defined), truncating/padding on the left. For alphanumeric fields, elements are mapped without conversion, truncating/padding on the right.	  	Lit (for literal values) Field (for internal fields) Source (for input elements)







Whether an item is identified with **Depends on** or **Repeats from** is controlled by the TRM: it uses **Depends on** for single input segments and **Repeats from** for repeating input segments or loops (X12) or groups (EDIFACT).







Edibasic Mapping

You can write Edibasic instructions to create variables and modify processing that would otherwise occur because of visual mapping. You write Edibasic instructions to override the default behavior of visual mapping for specific entities or objects, such as the document or wrapper, loops, segments, composite elements or elements.

EDI structures are defined as nested entities, so you have elements within composites, elements and composites within segments, segments, segments within loops, loops within loops, loops within documents or wrappers. We refer to these nested structures as levels: for example, document or wrapper level, loop level, segment level, or element level. Tabs allow you to enter your Edibasic instructions according to what you want to do at a given level for a specific entity or object.

There is a maximum of 6 tabs. You can perform certain operations at certain levels, and not at other levels. This is why you may not see all tabs at all times. Tabs are visible and available for your input only when it is appropriate at the level you have selected. For example, you do not create variables for element level processing, but you can create variables for all other levels. Therefore, you do not see the **Variables** tab for elements, but you do for the document or wrapper, loops, and segments. When you enter code on one of these tabs, an icon appears next to the mapped item as a visual cue that there is Edibasic code associated with the item. The following table shows you the relationship between the levels and the tabs that are visible for each:

Level	Tabs	Icon	Typical Use
Document or wrapper	Variables		To declare variables whose scope is valid for the entire document.
	Start		To initialize variables declared at this level. To enter Edibasic code that executes before any other code (e.g., to check a value to see if you want to continue mapping or not).
	Condition		To control candidacy for generation of the document based on data content. Instructions here execute before instructions on the Start page, before routing occurs for that document and before wrappers are generated, but after TRM routing has been determined.
Loop, repeating segment or repeating element	Variables		To declare variables whose scope is valid for all occurrences.
	Start		To initialize variables once for all occurrences.
	GetNext		To generate a specific occurrence of loop or segment that could not be done with visual mapping (e.g., creating multiple output segments or loops from a single input segment or loop).

Level	Tab	Icon	Typical Use
	Stop		To manually control resetting of occurrences after generation.
	Condition		To control candidacy for generation of specific occurrence of loop or segment based on data content.
Non-repeating segment or composite element	Variables		To declare variables whose scope is valid only for that segment or composite element.
	Start		To initialize variables for the segment or composite element.
	Condition		To control candidacy for generation of the segment or composite element based on data content.
Elements	MapEle		To control generation of an element by changing the input from what it would be with visual mapping, such as mapping from variables or modifying the data format.
	Condition		To control the candidacy for generation of the element based on data content.

IMPORTANT: There is currently a 32K limit on the size of compiled code for a single Edibasic method. If you find that you need to exceed this limit, you can create user exits to access user-defined subroutines and functions. For more information on user exits, refer to the *MW Translator User Exits Programming Manual*.

Sources for Output

You can directly relate input to output with visual mapping, and you can also enter literals and restrict the output occurrences to once, regardless of the number of times the input repeats. Typically, when you relate repeating structures on the input, you produce as many on the output. There are times that you want more control over what you create and how you create it. For instance, you might want to state some conditions where you test for values to make sure you create segments with valid data. That is when you would use Edibasic.

The purpose of the example included with the Workbench is to show you some of the mapping options available to you. As with all programming, you often have many options available to accomplish the same task. Our example shows you one possibility among many. Once you understand how mapping works, do not hesitate to try your own ideas.

Understanding Visual and Edibasic Mapping Techniques

You can perform the following set of instructions to get a feeling for visual mapping techniques. When you have finished these mini-exercise steps, you can delete the map you will have created.

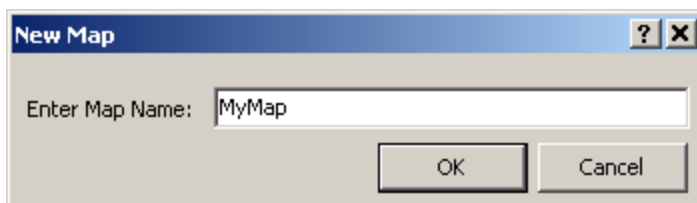
There are some basic principles behind both visual mapping and Edibasic. Once you understand these, you will better understand the strengths of each approach. First, we will create a small map and then apply basic visual and Edibasic techniques to create it.

Creating a Map to Learn Mapping Techniques

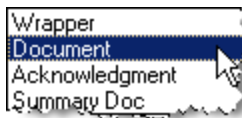
- 1 From the left pane, select **Maps**.
- 2 From the **File** menu, select **Add**, or right-click and select **Add** from the menu.


The **New Map** dialog box appears.

- 3 Enter the name of your practice map, such as **MyMap**.



- 4 Select **OK**.
The Mapping window appears.
- 5 In the header area, enter a description, **Learning Map**.
- 6 Select the map type, **Document**, from the list.




- 7 From the **Doc** tab in the left pane, select the **Select Source Doc** button .

The **Select Document** dialog box appears. The colored and underlined text allows you to double-click and jump directly to that document definition.

- 8 From the dialog box, select **X12, Example** in the upper pane and then the document **850** in the lower pane.

- 9 Select **OK**. The definition name appears in the header area and the segments and elements in the detail area in pane to the left.



- 10 From the header area in the right pane, select the **Select Destination Doc** button , which takes you to the **Select Document** dialog box.

- 11 From the **Select Document** dialog box select **Example, 1** in the upper pane and then the document **FPO** in the lower pane.

- 12 Select **OK**.

The definition name appears in the header area and the segments and elements in the detail area in pane to the left.



TIP: You should complete all of your definitions before you begin to map. If your definitions are incomplete, the Workbench displays what it has of the definition. For ease of use, you may partially define entities. If a document or wrapper that you entered looks incomplete or strange when it is displayed on the EDI Mapping window, make sure you go back and check the definition to see that it is complete, and not missing segments or elements.

Using Drag-and-Drop

- 1 In the left pane, left-click the input item (loop/group, segment, or element).
- 2 While holding down the button, move the cursor to the output item (loop/group, segment, or element) and release the mouse button.

A drag-and-drop icon will appear as you move the cursor. When you release the button, a check mark appears next to the mapped item, and a statement appears at the bottom of the output area indicating the type of mapping.

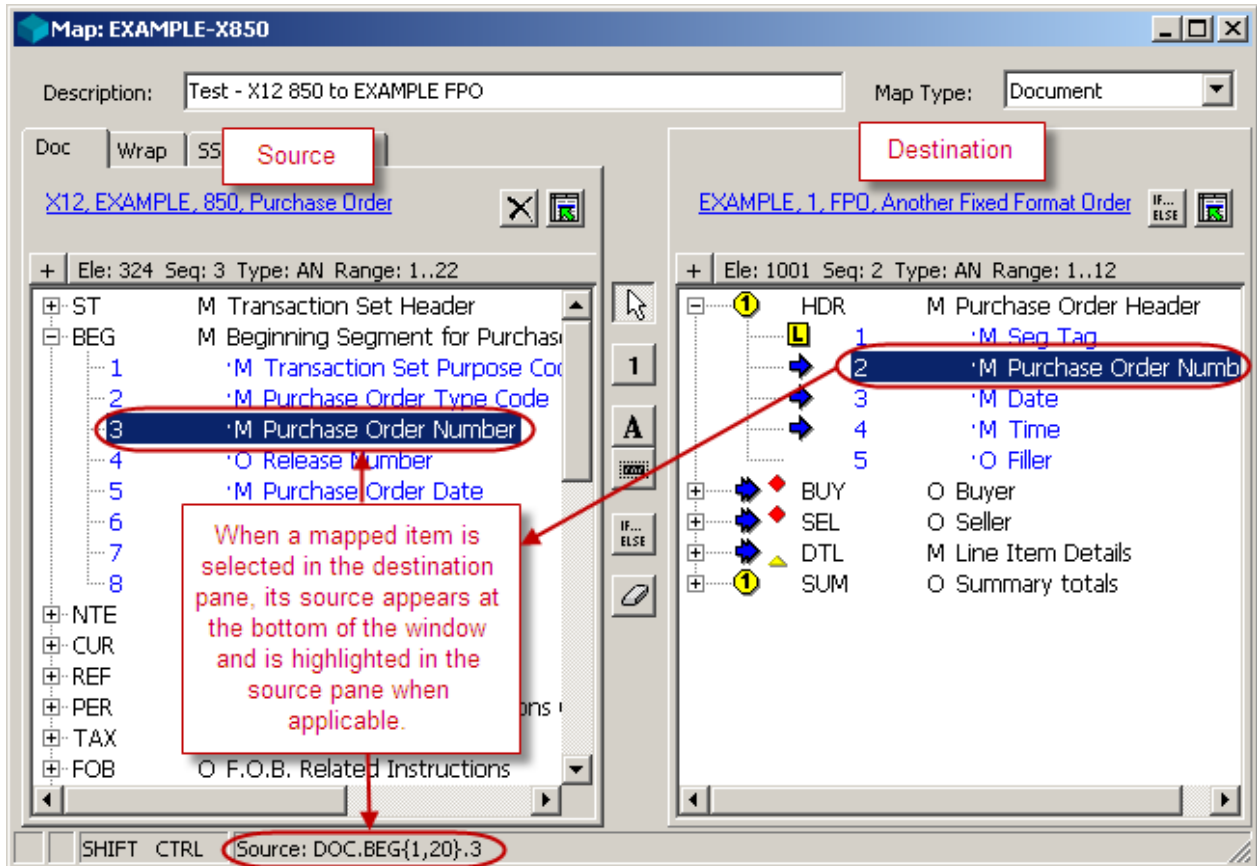
Visual Mapping Techniques

Typically, you use visual mapping when the input structure is not greatly different from the output structure.

Visual mapping can be used for the following input and output relationships:








- single segment to single segment
- single segment to repeating segment
- single segment to loop/group
- repeating segment to repeating segment
- repeating segment to loop/group
- loop/group to repeating segment
- loop/group to loop/group
- single element to single element
- single element to repeating element

The source and destination definitions appear in an outline format. When first displayed, you see the highest levels of the definition, the segments outside of loops/groups and the highest-level loops/groups. Double-click items to see lower levels of detail, or click the expand/collapse button to show all items or show only the highest level items. Mapped destination items are linked to source items where possible. Whatever the source, it appears at the bottom of the source pane. Items that cannot be expanded further are preceded by a dot and displayed in blue.



Other visual clues about the source of a mapped item are shown in the following table:

Source Statement	Icon	Description
Once	①	The selected segment, loop, composite element or simple element is to be mapped only once, regardless of the number of times the input repeats, even when there is no input.
Depends on	➔	The selected non-repeating entity such as a segment, loop or element is to be mapped not at all or once based on the existence of the identified source, which can be repeating or non-repeating.

Source Statement	Icon	Description
Repeats from		The selected repeating entity such as a segment or loop is to be mapped based on the number of repetitions of the identified source.
Source		Identifies the source of the element. For the syntax of the input location name, refer to the topic, Data Element Names (on page 698).
Literal		The element is mapped from the literal value displayed within quotes (maximum 43 characters).
Field		The element is mapped from one of the internal fields, a list of which is available from the Select Field dialog box.
Method		The item is mapped using Edibasic MapEle or GetNext methods.
		The item has Edibasic code on a Variables, Start or Stop tab.
		The item has Edibasic code on a Condition tab.

Visual Mapping Default Behavior







The Workbench makes certain assumptions when you use visual mapping. If you know what these are, you will be able to determine if or when you will want to use Edibasic. The effect of visual mapping is as follows:

- When you map an element or segment, a default mapping of ONCE is assigned to its segment or loop/group, respectively, if it has not been already explicitly mapped.
- When you explicitly map segments or loops/groups from input segments or loops/groups, the number of repetitions of the output is controlled by the number of repetitions of the input. The TRM creates as many occurrences of the output as there are of the input.
- For repeating entities (loops/groups, segments), the repetition level for the input is increased by 1 for the next occurrence.
- For nested entities (loops/groups, segments), the scope of the entities depends on the nesting level.

To perform other visual mapping techniques from the Map window using the tool buttons, you would click the appropriate button, then click the target item (segment, loop/group, or element), make a selection or enter text if required, then click **OK**.

Using Mapping Tools


You access these tools from the Map window using buttons shown in the table.

Button	Description	Effect
	(normal cursor)	Normal selection.
	Once	Creates only one occurrence of the output, regardless of how many times the input occurs.
	Literal	Maps a literal value to an output element (maximum 43 characters).
	Internal Field	Maps the contents of an internal field to an output element.
	Edibasic Method	Provides access to Edibasic mapping
	Erase	Erases a previously mapped instruction (does not delete the item).

These additional visual mapping techniques allow you to satisfy many mapping requirements without using Edibasic.

To Use Visual Mapping Buttons

The following figures show you the sequence of steps you would use for one of these other techniques, mapping literals.

- 1 Click the button associated with the tool, such as the **Literal** button .
- 2 Click the destination element, such as **HDR.1**.
When you release the mouse button, the Mapping Input window appears.
- 3 What you do next depends on the tool. For this tool, enter text up to 43 characters and select **OK**.



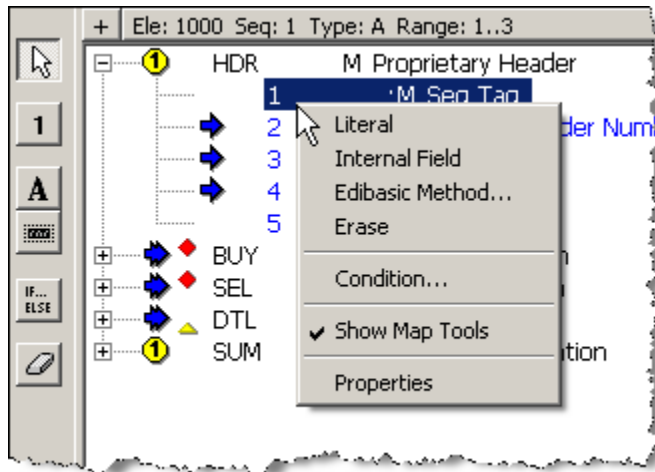
TIP: To enter a string longer than 43 characters, use the Edibasic method MapEle.

To Use the Drop-Down Menu

- 1 Click the **Cursor** button, and right-click the destination item, such as **HDR.1**.

A menu appears, with various options, depending on the level of the item being mapped. The **Edibasic Method** and **Condition** options are for Edibasic, which we will discuss later.

NOTE: You can turn the display of the map buttons in the center on or off by unchecking or checking the **Show Map Tools** option.



2 Select the technique you want.




Edibasic Mapping Techniques

You will need to use the Edibasic editor for the following:

- To enter literal string values longer than 43 characters
- To map dissimilar structures, such as a nested loop/group to a non-nested loop/group, or separate segments to a loop/group
- To use an expression to calculate the value of an element
- To use conditional expressions to determine when to create output
- To convert formats other than DateTime and numeric from input to output. (The TRM automatically converts formats if a DateTime or numeric element has a format defined as part of its type.)
- To pass information among documents or interchanges within the same input stream by using global variables

Since you can write Edibasic methods at various levels (document, loop/group, segment, element), you must know for which level or item you wish to write a method. To write Edibasic methods, you must access the Edibasic Editor from the Map window.

There are different techniques for the different levels:

Level	To Write a Method
Header: Document	Select the Document Variables and Methods button  , next to the Select Destination Doc button 
Detail: Loop/group, Segment, Element	Select the Edibasic Method button  in the middle of the window, and then select the target item so that it is highlighted – or – Select the target item so that it is highlighted, and click the right mouse button. Select Edibasic Method , Variables , or Condition from the drop-down menu.


NOTE: Edibasic selections are available when they are appropriate. For example, you do not see the **Variables** selection when you select an element.

For a working example with some Edibasic techniques refer to the Example-X850 map that is used with the test example in the Workbench.

Deleting Your Learning Map

Now that you have finished these mini-exercises, you can delete it if you wish. To delete a map:

- 1 In the left pane, select the **Maps** folder.
- 2 In the right pane, select the map you just created, **MyMap**.

- 3 From the toolbar, select the **Delete** button .

– or –

From the **File** menu, select **Delete**.

– or –

Right-click and select **Delete** from the menu.

- 4 Select **OK** from the Confirm dialog box.

A progress box appears, and when the task is complete, you return to Data Explorer.

Task 2. Creating Proprietary Maps

You will need maps for both the wrapper and the document. You will have to create a map for your proprietary wrapper if you have never done so before. Usually this is a one-time task. You will also have to create a map for your document.

Overview of Creating Proprietary Maps

Each time you create or change a map that you will need to use during processing, you must generate a new text file that includes the mapping instructions. You can generate the maps with everything else at the end of your entire configuration process, but it is better to generate them after you define the entity, so you can correct any problems at an early stage.

Although you can create either the wrapper or the document map first, this discussion of tasks is as follows:

- 1 Create a map for your proprietary wrapper.
- 2 Generate a text file for your proprietary wrapper map.
- 3 Create a map for your proprietary purchase order.
- 4 Generate a text file for your proprietary purchase order map.
- 5 Print map reports.

Creating Maps Task Tree

The following figure shows you the primary tasks associated with creating maps. Although not listed as a sub-task, at some point you must also generate a text file for the map, and you should print a map report.

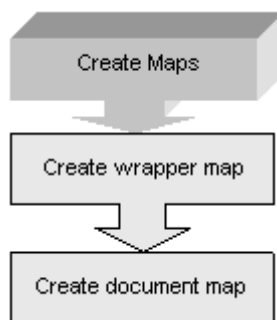


Figure 1. Creating Maps Task Tree

Task 2.1. Creating a Wrapper Map

Maps for wrappers typically do not use any source document as input. Most input comes from literals, so the mapping can be relatively simple. Maps for the public standard wrappers are loaded during installation, which you can review from the Maps folder in Data Explorer. Since the wrapper definitions for public standards are typically more complex than the wrappers for a proprietary standard, their maps will be larger.

Task Tree to Create a Wrapper Map

The following sub-tasks support the task to create a wrapper map. Although not listed as a sub-task, at some point you must also generate a text file for the map, and you should print a map report.

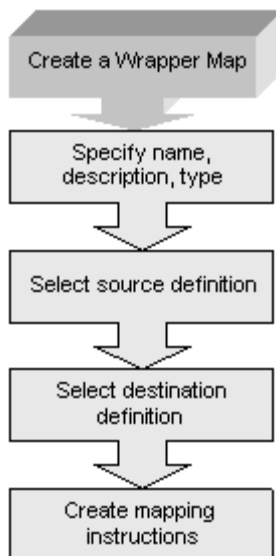


Figure 2. Sub-tasks to Create a Wrapper Map (Creating Maps Task Tree)

Task 2.1.1. Specifying the Map Name, Description, and Type for the Wrapper

When you add a map, you specify the map name, which appears in the title bar of the Map window. On the Map window, you enter a description, and then select the map type, **Wrapper**.

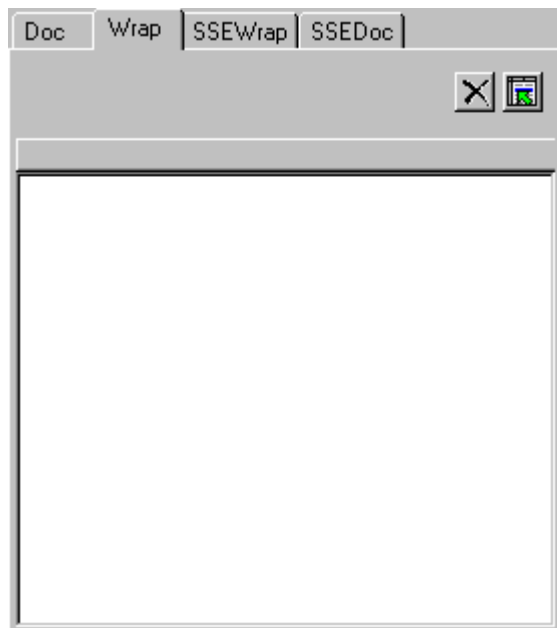
IMPORTANT: Once you identify a map as a wrapper map, you cannot redefine it as another map type. If you wanted to create a map type other than a wrapper map, you must create a new map.




Task 2.1.2. Selecting the Source Definition

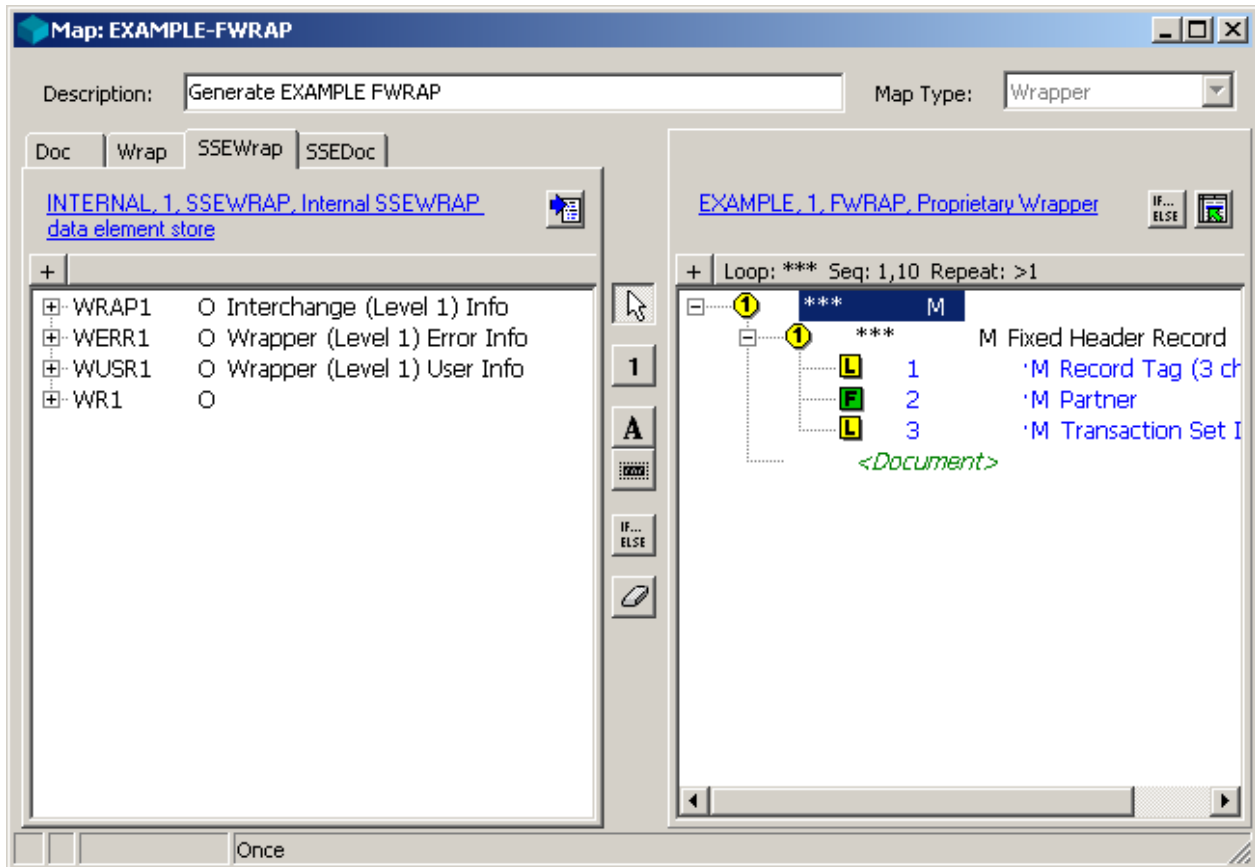
There are four tabs in the source area of the Window. You cannot map from a document, and typically, you map from literals or internal fields. During mapping the internal fields contain information that has been extracted partially from incoming documents and partially from configurations.

In the case of the EXAMPLE-FWRAP map you notice that there is no source selected.



Task 2.1.3. Selecting the Destination Document

You select the destination location by choosing the **Select Destination Doc** button . When you expand the one wrapper segment of the EXAMPLE-FWRAP map you notice that there are only three elements. The Record Tag and Transaction Set ID are mapped from literals, and the Partner is mapped from the internal field, **Send Partner ID**. This will not be the same ID that was the Sender ID in the incoming wrapper, because the partnership definitions have specified another ID. You can see how this is done in the section, *Defining Partners* (on page 243).



Task 2.1.4. Entering Mapping Instructions for the Wrapper

The EXAMPLE-FWRAP map uses all visual mapping techniques. You can quickly review the mapping for each entity, loop/group, segment, composite or element by selecting the entity to see the mapping at the bottom left of the Map window. For a discussion of the actual techniques of visual mapping, refer to the section, *Understanding Visual and Edibasic Mapping Techniques* (on page 159).

Task 2.1.5. Generating the Wrapper Map

To generate the wrapper map from the Wrapper window, select **Map** from the **Generate** menu. If you have closed your Wrapper window, you can select your map in the right pane and then select **Map** from the bottom of the **Generate** menu.

Task 2.2. Creating a Document Map

Maps for documents typically use a source document, and possibly a wrapper, as input.

Task Tree to Create a Document Map

The following sub-tasks support the task to create a document map. Although not listed as a sub-task, at some point you must also generate a text file for the map, and you should print a map report.

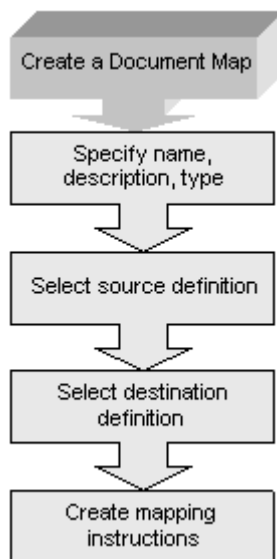


Figure 3. Sub-tasks to Create a Document Map (Creating Maps Task Tree)


Task 2.2.1. Specifying the Map Name, Description, and Type for the Document

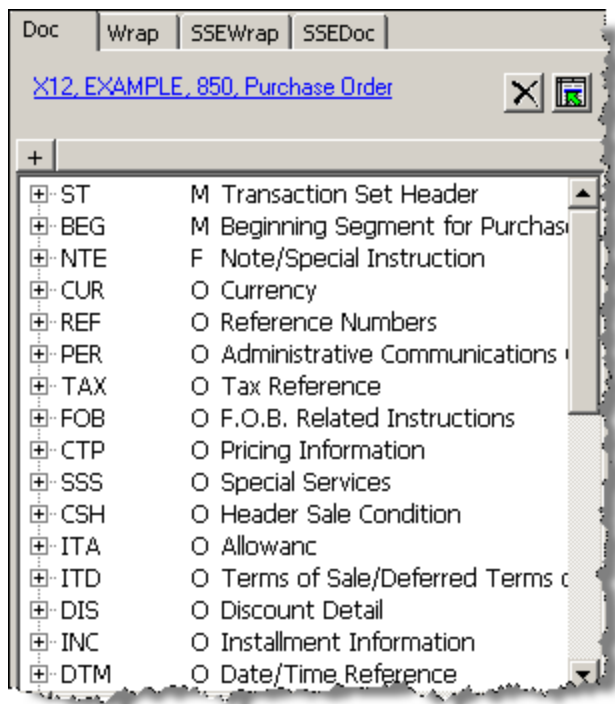
When you add a map, you specify the map name, which appears in the title bar of the Map window. You enter a description, and then select the map type, **Document**.

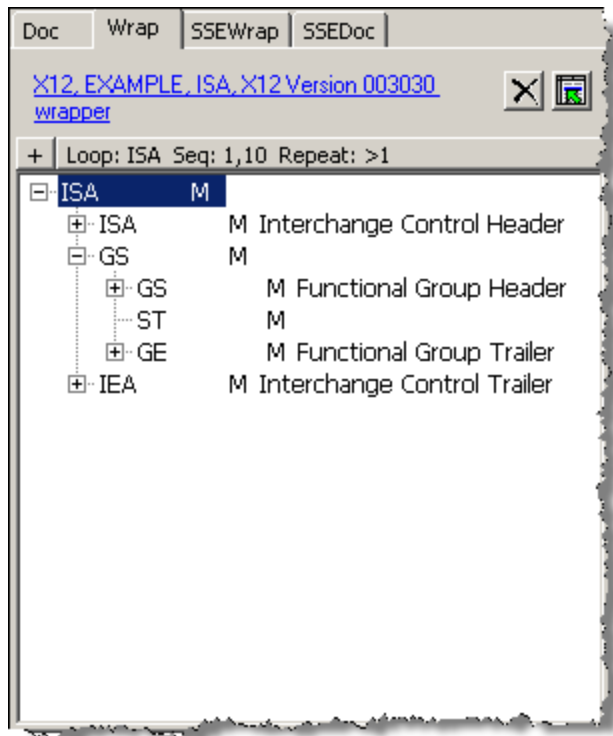


Task 2.2.2. Selecting the Source Definition


There are four tabs in the source area of the Window. You can map from a document, a wrapper, and even from the status, statistics and errors tabs (**SSEDoc** and **SSEWrap**). You can also map from literals and internal fields. During mapping the internal fields contain information that has been extracted partially from incoming documents and partially from configurations. You may also use Edibasic to create some of your information.

You choose a source by selecting the appropriate tab and then the **Select Source Doc** button . Note that in the EXAMPLE-X850 map there are sources selected for both the **Doc** and **Wrap** tabs.





Task 2.2.3. Selecting the Destination Definition

You select the destination location by choosing the **Select Destination Doc** button .

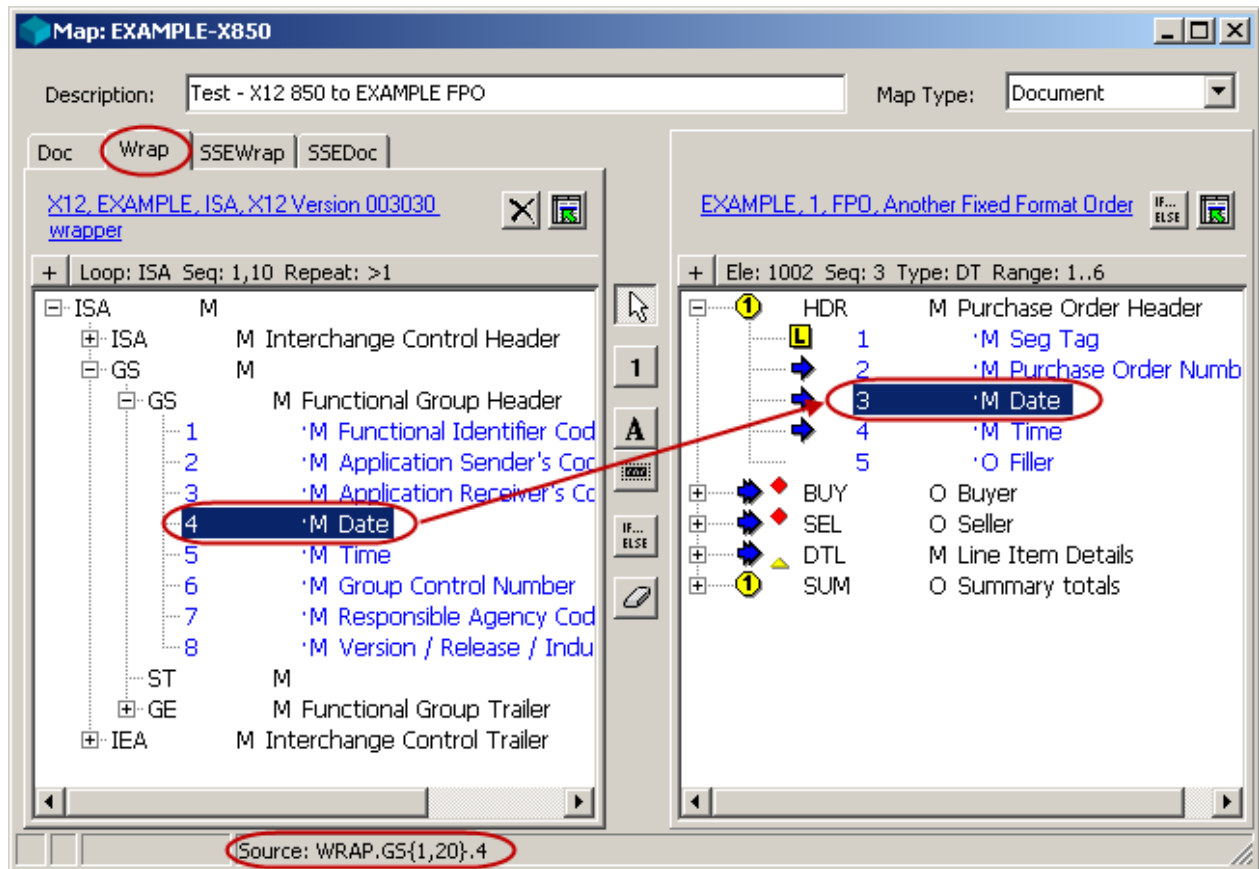


Task 2.2.4. Entering Mapping Instructions for the Document

To mention a few points about the EXAMPLE-X850 map, you will notice that it uses drag-and-drop mapping from the source document. For an example and an explanation of the mapping symbols, refer to the topic, *Visual Mapping Techniques* (on page 161).

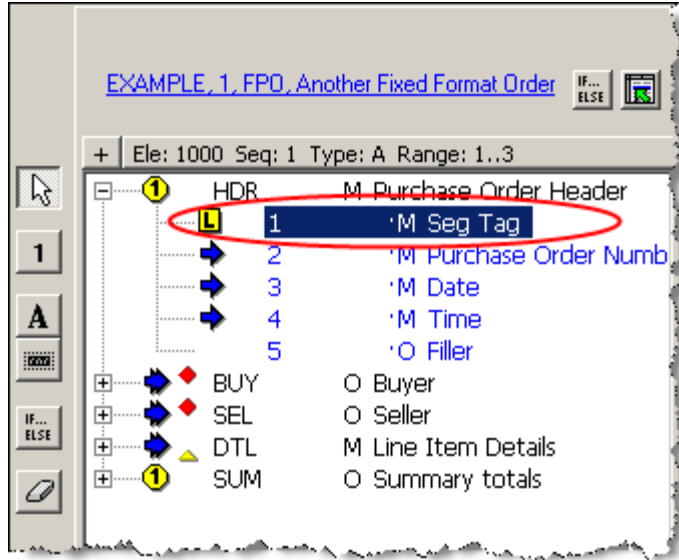
Example of Mapping from the Wrapper

The example also uses drag-and-drop mapping from the wrapper.



Example of Mapping from a Literal

The example maps from literals.

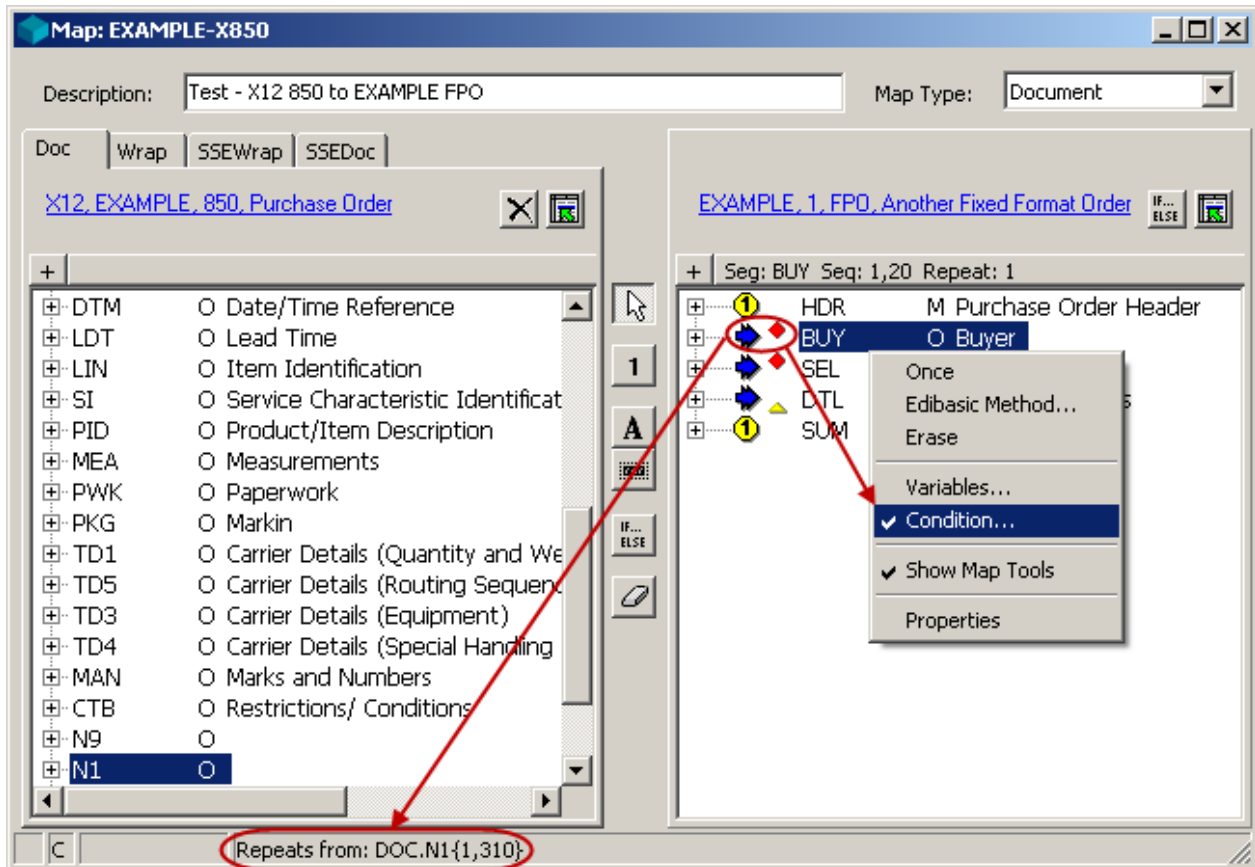


Example of Mapping Using Conditions

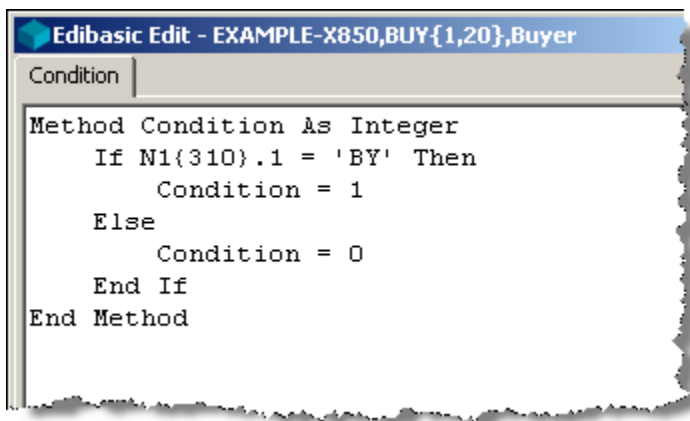
The example uses a combination of drag-and-drop and Edibasic:

- It maps from the N1(name and address) loop to the BUY single segment, to access all occurrences of the loop. Note that it maps from the N1 loop, not the N1 segment, which occurs only once within an instance of the loop.
- It then uses an Edibasic condition, indicated by the condition symbol, **◆**, and the **C** in the lower left corner, to select the correct occurrence, the buyer information.

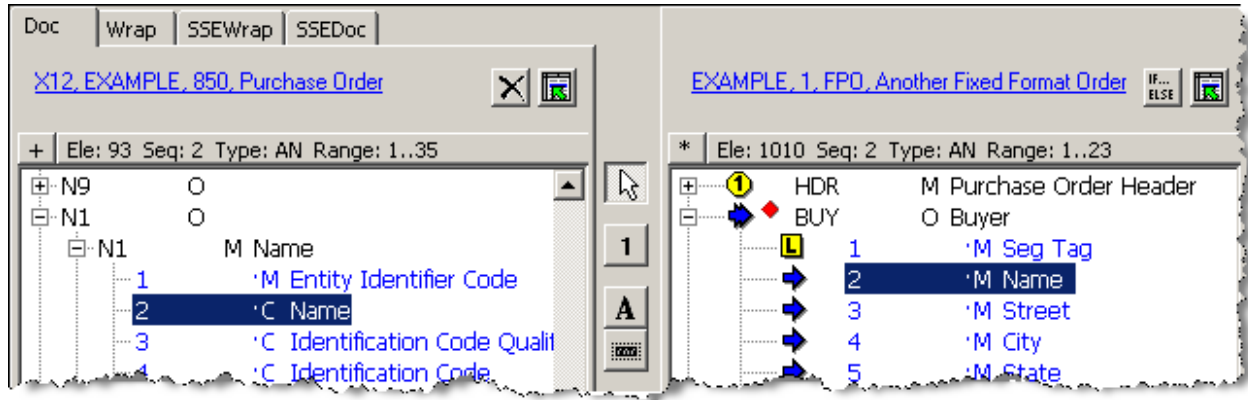
Since the input contains two instances of the N1 loop, one that contains buyer information and one that contains seller information, the condition checks all occurrences and selects the ones that match the specified condition.



Since the input contains two instances of the N1 loop, one that contains buyer information and one that contains seller information, the condition checks all occurrences and selects the ones that match the specified condition. In this case it is looking for a code of **BY** in the N1.1 element before it maps the information to the BUY elements.



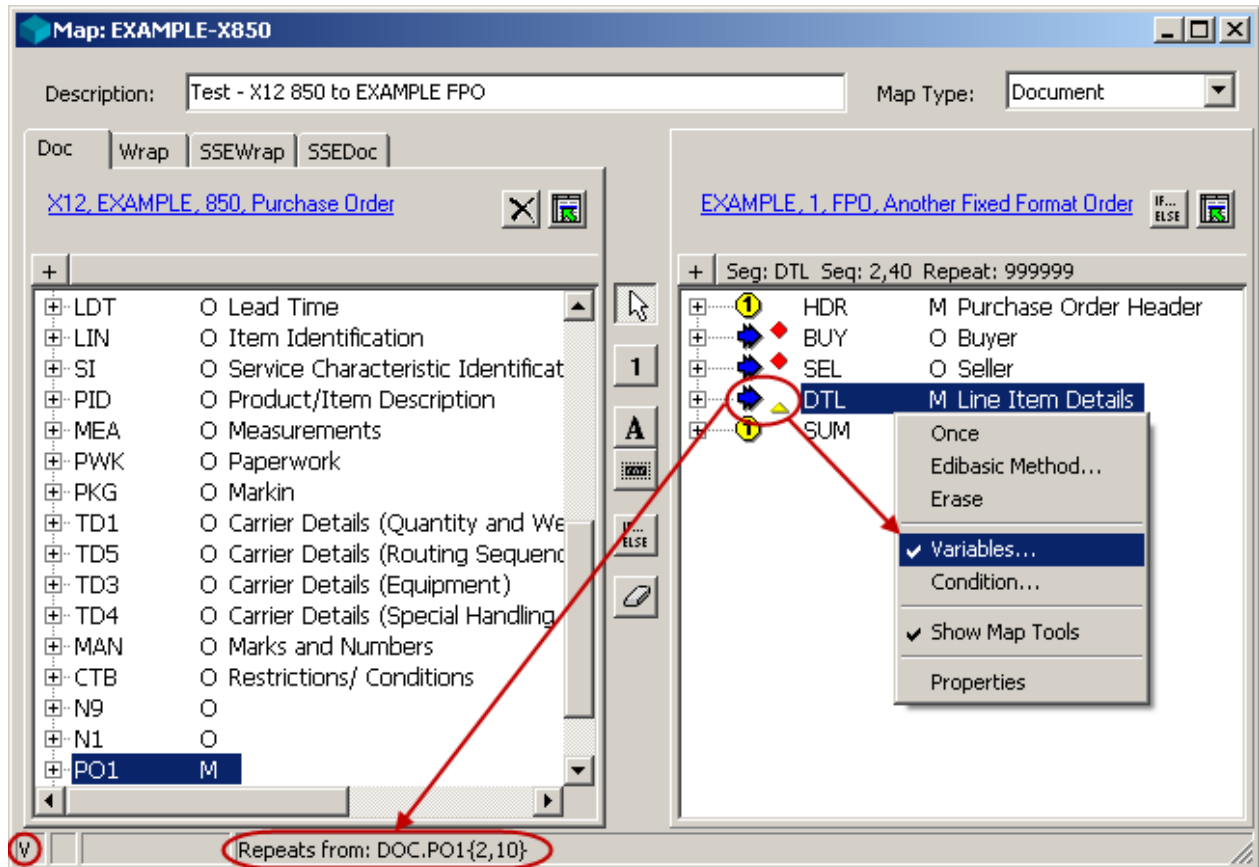
You can use drag-and-drop to map the elements within the BUY segment. The element output is created only when the condition on the segment is satisfied.



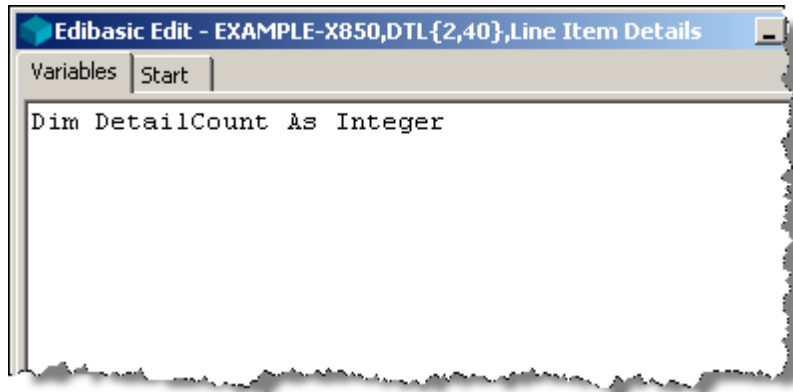
Example of Mapping Using Variables

In another case, for the DTL segment, the map creates and initializes a variable at the segment level, indicated by the variable symbol, \blacktriangle , and the **V** in the lower left corner. It then uses the variable to create line numbers for each of the line items, because there were no line numbers it could use from the source document.

First, the DTL is mapped from the PO1 loop, to generate all the occurrences of the line items. Note that there is no condition associated with the DTL repeating segment, so the TRM will generate as many occurrences of the output as there are of input.

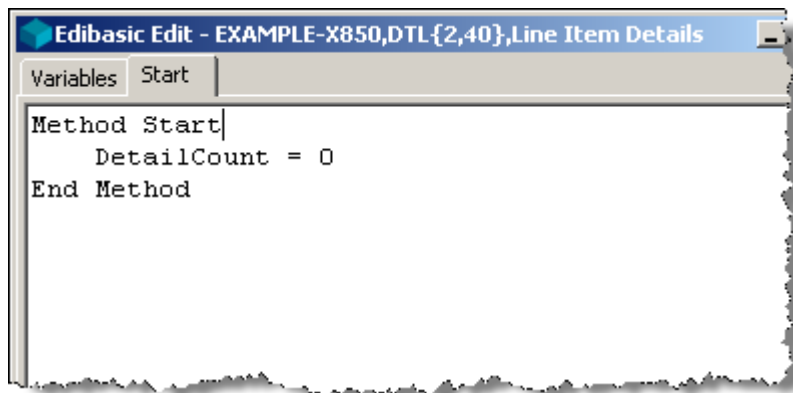


The variable is defined and initialized at the segment level, so it will be created once for all occurrences and can then be incremented as the TRM processes each occurrence.



The screenshot shows a window titled "Edibasic Edit - EXAMPLE-X850,DTL{2,40},Line Item Details". The window has two tabs: "Variables" and "Start". The "Variables" tab is selected, and the text "Dim DetailCount As Integer" is visible in the editor area.

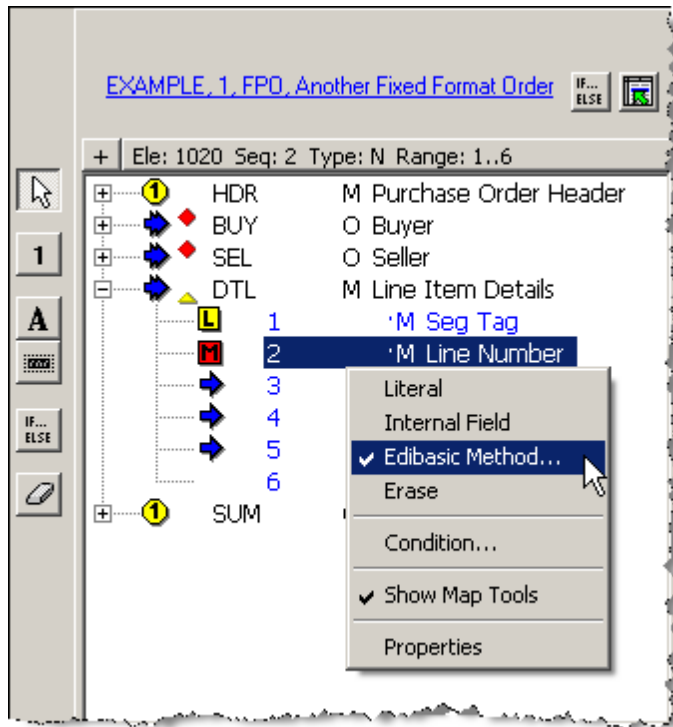
```
Dim DetailCount As Integer
```



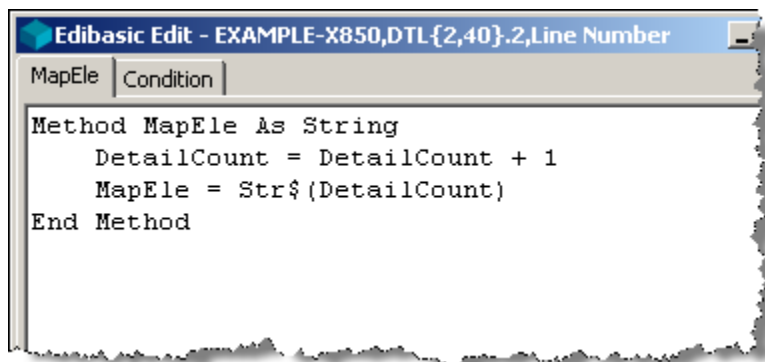
The screenshot shows the same window as above, but now the "Start" tab is selected. The text in the editor area shows a method named "Start" that initializes the variable "DetailCount" to 0.

```
Method Start|
    DetailCount = 0
End Method
```


The TRM increments the variable as it generates the line item element based on the Edibasic method.



The MapEle method places data as a string in the element. Notice that the variable, DetailCount, was created as an integer in order to perform the addition operation. The integer value must then be converted to a string before it is stored in the element, which is done using the Edibasic function Str\$.



For specific information about Edibasic operators, methods, and functions, refer to the Edibasic reference material in the online help.



Str\$ function

Description Returns a string representing the value of a numeric expression, x.

Syntax **Str\$ (x)**

Valid Values x any Edibasic expression that returns a numeric expression.

See Also [Format\\$, Val](#)

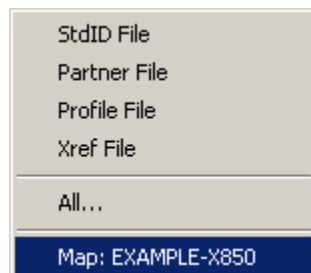
Example

The following example uses the **Str\$** function to convert a numeric expression returned by the **Count** function, which contains the number of occurrences of the N1.1 element.

```
MapEle = Str$(Count(N1.1))
```




Task 2.2.5. Generating the Document Map

To generate the document map from the Document window, from the **Generate** menu, select **Map**.



Task 2.3. Printing Map Reports






You can print reports of your maps by selecting the appropriate map in the right pane. From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file
- The **Print Preview** button  to print reports to the screen
- The **Print to PDF** button  to print to a PDF file

MW Translator Workbench Map Report

EXAMPLE-FWRAP, Generate EXAMPLE FWRAP
(Wrapper Map)

Destination Wrapper: EXAMPLE, 1 FWRAP, Proprietary Wrapper

	Once		***{1,10} Loop
	Once		***{1,10} Fixed Header Record
	****		***{1,10}.1 Record Tag (3 character)
<Send Partner ID>			***{1,10}.2 Partner
"FPO"			***{1,10}.3 Transaction Set ID

The following figure shows only part of the EXAMPLE-X850 map.

MW Translator Workbench Map Report

**EXAMPLE-X850, Test - X12 850 to EXAMPLE FPO
(Document Map)**

Destination Document: EXAMPLE, 1 FPO, Another Fixed Format Order
 Source Document: X12, EXAMPLE 850, Purchase Order
 Source Wrapper: X12, EXAMPLE ISA, X12 Version 003030 wrapper

<p>Once</p> <p>"HDR" DOC.BEG{1,20}.3 WRAP.GS{1,20}.4 WRAP.GS{1,20}.5</p> <p>Repeats from DOC.N1{1,310} Method Condition</p> <pre>Method Condition As Integer If N1{310}.1 = 'BY' Then Condition = 1 Else Condition = 0 End If End Method</pre>	<p>① HDR{1,10} Purchase Order Header</p> <p>Ⓛ HDR{1,10}.1 Seg Tag ➡ HDR{1,10}.2 Purchase Order Number ➡ HDR{1,10}.3 Date ➡ HDR{1,10}.4 Time</p> <p>➡ BUY{1,20} Buyer ● BUY{1,20} Buyer</p> <p>Ⓛ BUY{1,20}.1 Seg Tag ➡ BUY{1,20}.2 Name ➡ BUY{1,20}.3 Street ➡ BUY{1,20}.4 City ➡ BUY{1,20}.5 State ➡ BUY{1,20}.6 Zip</p>
---	---

Advanced Mapping Techniques

People who do mapping should know how to access input data that repeats or is embedded in a hierarchical structure.

There are two types of MW Translator mapping techniques, either of which you can use to map repeating and embedded information:

- Visual, and
- Edibasic


These techniques work in the same way, but you have more control using Edibasic. Therefore, mappers should use Edibasic when visual mapping does not meet their needs.

One task of the mapper is to access the information from the input document in the Data Element Store (DES). Sometimes the data is repeating or embedded several levels deep within a series of loops. For repeating and embedded entities, the TRM uses a single context list to keep track of the levels and multiple occurrences at each level for the current input document. Non-repeating, non-embedded segments, also called global segments, use a separate context list that is null. The scope of a segment is reflected in the context list.

The context list appears in the DES listing on the processing report. The TRM uses one context list for the entire processing cycle, which it maintains based on mapping instructions. When mappers use Edibasic to maintain the context list, they control the levels and occurrences themselves.

To Display the Contents of the Data Element Store

To view the context list for a particular input segment and its elements, users may display the contents of the Data Element Store (DES) on the processing report. To display the contents of the DES, do the following from the MW Translator Workbench:

- 1 From the **File** menu, select **Options**.
- 2 From the Modify Options window, select the **General** tab.
- 3 Check **Always print reports**.
- 4 Optionally check any of the following boxes:
 - a) **Print DES** prints the locations and the contents of the data element store, which contains the input document, wrapper, and status, statistics, and errors to be reported in backward control documents.
 - b) **Print File Names** prints the names of the files used or sought during processing.
- 5 *Run a test* (on page 771).
- 6 Click the processing report icon, . Its file should have the same name as the input file for the test.

NOTE: If there are two report icons, the processing report is the second one.

Understanding the Context List in the Data Element Store

The levels and occurrences comprise the context list for repeating and embedded segments. Each level is represented by a position in the list. The number of items in the context list represents each of the levels. For example, a context list of 1 1 1 represents three levels. The current occurrence is the number indicated for each level.

Consider the following DES dump. Note that segments that are not embedded, such as S1, have no context list. Remember that non-repeating segments that are not embedded in any loops are also called global segments. Only repeating segments and segments embedded in loops use the context list, such as S2 through S6.

NOTE: The DES contains only segments that appear in the input file. Segments not in the input file are not listed. In addition, it shows only the segments, without the loop or repeating segment definitions, which are implied by the context level(s) and repetition number(s) listed for each segment.

Dump of Data Element Store: DOC1

Segment: S1(1)

1 'S1'
2 'INFO'

Segment: S2(2)

Occurrence: 1

1 'S2'
2 'INFO'

Segment: S6(3)

Occurrence: 1 1

1 'S3'
2 'INFO'

Occurrence: 1 2

1 'S3'
2 'INFO'

Segment: S3(4)

Occurrence: 1 1

1 'S3'
2 'INFO'

Segment: S4(6)

Occurrence: 1 1 1

1 'S4'
2 'INFO'

Occurrence: 1 1 2

1 'S4'
2 'INFO'

Segment: S5(8)

Occurrence: 1 1 1 1

1 'S5'
2 'INFO'

Occurrence: 1 1 2 1

1 'S5'
2 'INFO'

End of Dump of Data Element Store: DOC1

The following table explains the information contained in a basic Data Element Store dump that appears on a processing report. The types of information display as follows:

Label	Type of Information	Sample Value
Dump/End of Dump of Data Element Store	Document name	DOC1
Segment:	Segment Tag and (absolute segment sequence number)	S6 (3)
Occurrence:	Context List (Levels and Occurrences)	1 1
None	Element sequence number	2
'....'	Value	INFO

Level in DES Versus Level in Document Configuration

You should make a distinction between levels as cited in the DES and those as cited in the document configuration. Compare the DES context list with the structure of the document in the Document window.

The levels given in the DES, shown in the preceding figures may not be the same as the levels listed in the Level column on the Document window. For example, the level of the first S6 in the DES is 2, i.e. 1 1, whereas the Level for the first S6 in the document definition is 1. To understand how to use the context list, you must use the levels as cited in the DES.

The screenshot shows a window titled "Document: PROPSTD, 1.0, PROPDO" with a description of "Proprietary File Invoice". Below the description is a table with columns: L/S, Level, Area, Seq, Tag, Description, Rqmt, and Max Occ. The table lists various segments and their levels, with lines indicating the structure of the document configuration.

L/S	Level	Area	Seq	Tag	Description	Rqmt	Max Occ
S	0	1	10	S1	Proprietary File Availment/Payment	M	1
L	0	1	20	S2	Loop	M	>1
S	1	1	20	S2	Proprietary File Invoice	M	1
S	1	1	30	S6	Proprietary File Discrepancy	0	10
L	1	2	40	S3	Loop	0	>1
S	2	2	40	S3	Proprietary File Purchase Order	0	1
S	2	2	50	S6	Proprietary File Discrepancy	0	10
L	2	2	60	S4	Loop	0	>1
S	3	2	60	S4	Proprietary File Purchase Order Line	0	1
S	3	2	70	S6	Proprietary File Discrepancy	0	10
L	3	2	80	S5	Loop	0	>1
S	4	2	80	S5	Proprietary File Purchase Order SubLine	0	1
S	4	2	90	S6	Proprietary File Discrepancy	0	10

Understanding Levels in the Context List

The first segment, S1, does not use the context list, because it does not repeat and it is not embedded in any other loops. In this example, there are four levels of loops in the definition: S2 is the highest level, and contains loop S3, which contains loop S4, which contains loop S5. The following table shows the context list for the first occurrence of each of the input loops and repeating segments listed in the preceding DES example.

ID	Level (Document Definition)	Access Level (Mapping)	Context List Example from DES
S2 Loop	0	1	1
S2 Segment	1	1	1
S6 Segment	1	2	1 1
S3 Loop	1	2	1 1
S3 Segment	2	2	1 1
S4 Loop	2	3	1 1 1
S4 Segment	3	3	1 1 1
S5 Loop	3	4	1 1 1 1
S5 Segment	4	4	1 1 1 1

The next table shows the context list of segments that have more than one occurrence, and occur in the input: S6, S4 and S5. Segment S4 occurs twice within the S4 loop. The segment S5 occurs twice, once for each occurrence of S4. The segments in this table are in a different order than the order in which they appear in the DES on the processing report to emphasize how they appear in the input file.

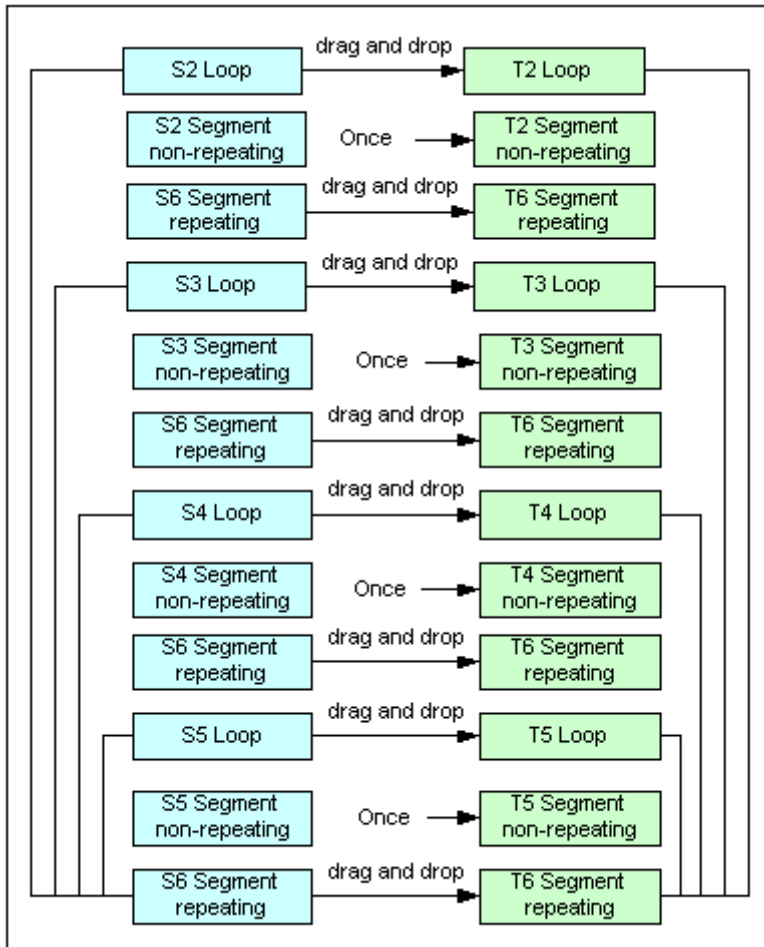
ID	Access Level (Mapping)	Occurrence	Context List Example
S2 Loop	1	1	1
S2	1	1	1
S6	2	1	1 1
S6	2	2	1 2
S3 Loop	2	1	1 1
S3	2	1	1 1
S4	3	1	1,1,1

S5	4	1	1,1,1,1
S4	3	2	1,1,2
S5	4	1	1,1,2,1

Effect of Visual Mapping on Context List

The purpose of visual mapping is to avoid the complexity of maintaining the context list. Visual techniques work well when the input structure and output structures are similar. When you understand how the TRM responds to visual mapping, you will understand how to use Edibasic functions to correctly access repeating and embedded entities.

Consider mapping the preceding input structure to a similar output structure. The following diagram is an overview of the visual mapping techniques used to map loop S2 through loop S5. The S1 segment, since it is a global segment, does not use the context list, so we begin with the S2 loop.

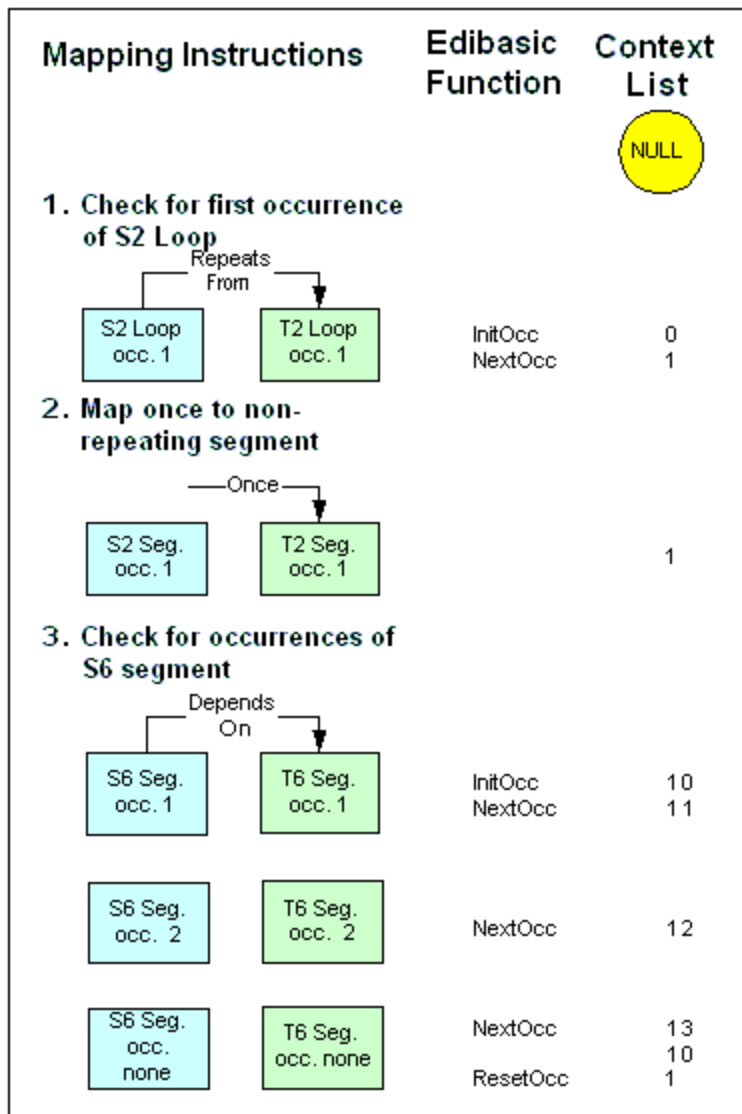


Overview of Visual Mapping Techniques for Similar Structures

In the following set of figures, the input is on the left and the output appears to its right. To the right of the output are the equivalent Edibasic functions that the TRM uses to maintain the list, e.g. **InitOcc**, **NextOcc** and **ResetOcc**. The functions are followed by the values in the context list as the TRM processes each mapping instruction listed here in numeric order.

The next figure is an overview of the effect of visual mapping on the first occurrence of the S2 loop. Note the following:

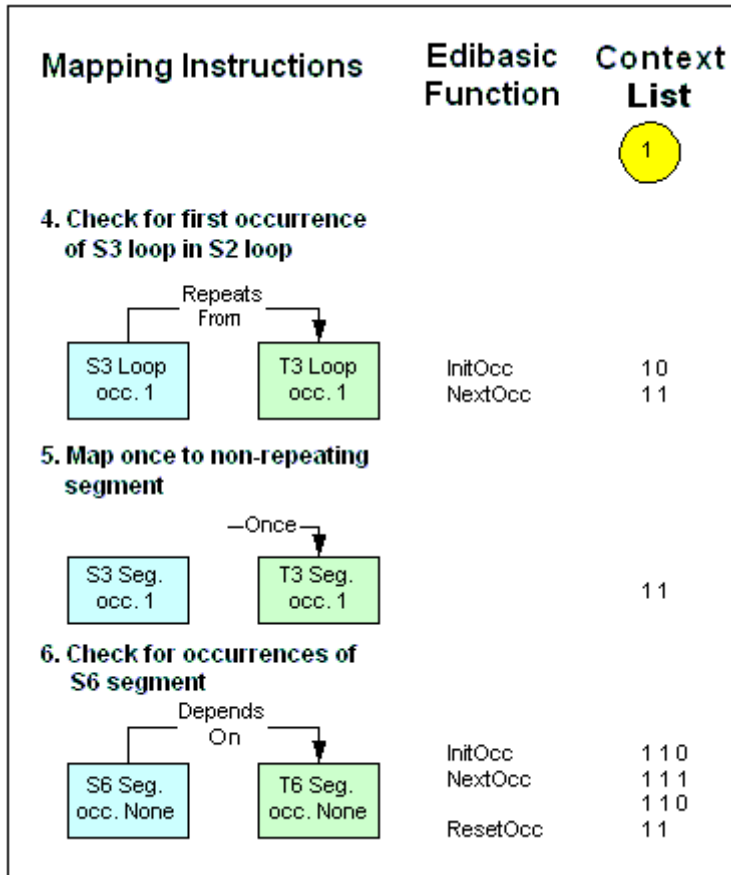
- The context list always begins with a null value
- Once mapping does not affect the context list
- **InitOcc** process sets the level and initializes it to zero
- **NextOcc** process checks for additional occurrences by adding to the level number until it finds no more, when it zeros the level
- **ResetOcc** process returns context list to what it was prior to the last **InitOcc**
- After processing the first occurrence of the S2 loop, the context list is 1



Effect of Mapping Equivalent Input and Output Structures on Context List

The next figure shows the effect of visual mapping on the first occurrence of the S3 loop. Not the following:

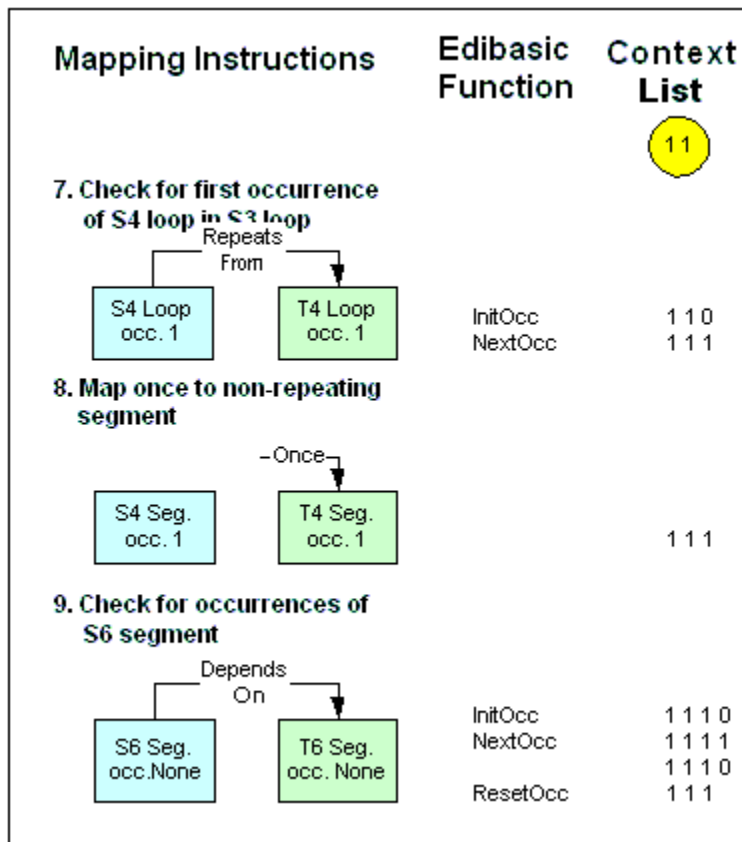
- After processing the first occurrence of the S3 loop, the context list is 1 1



Effect of Mapping Equivalent Input and Output Structures on Context List, cont.

The next figure shows the effect of visual mapping on the first occurrence of the S4 loop. Note the following:

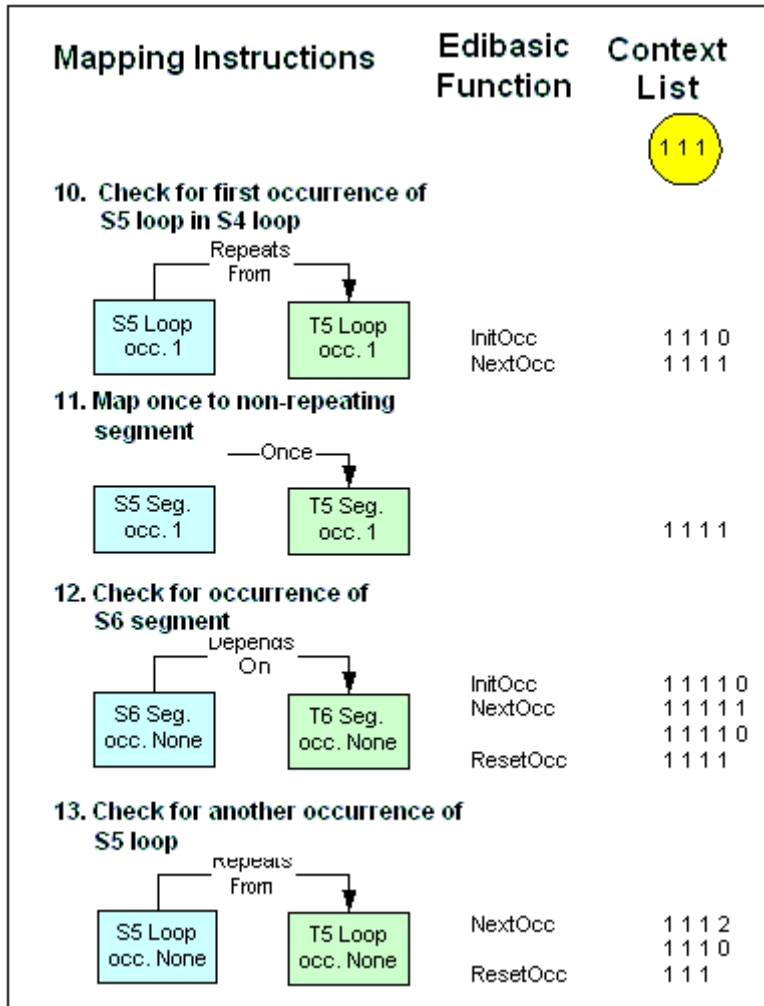
- After processing the first occurrence of the S4 loop, the context list is 1 1 1
- While processing occurrences of the S6 repeating segment, the context list is 1 1 1 1
- After finding no occurrences of the S6 segment, the context list reverts to 1 1 1



Effect of Mapping Equivalent Input and Output Structures on Context List, cont.

The next figure shows the effect of visual mapping of the first occurrence of the S5 loop. Note the following:

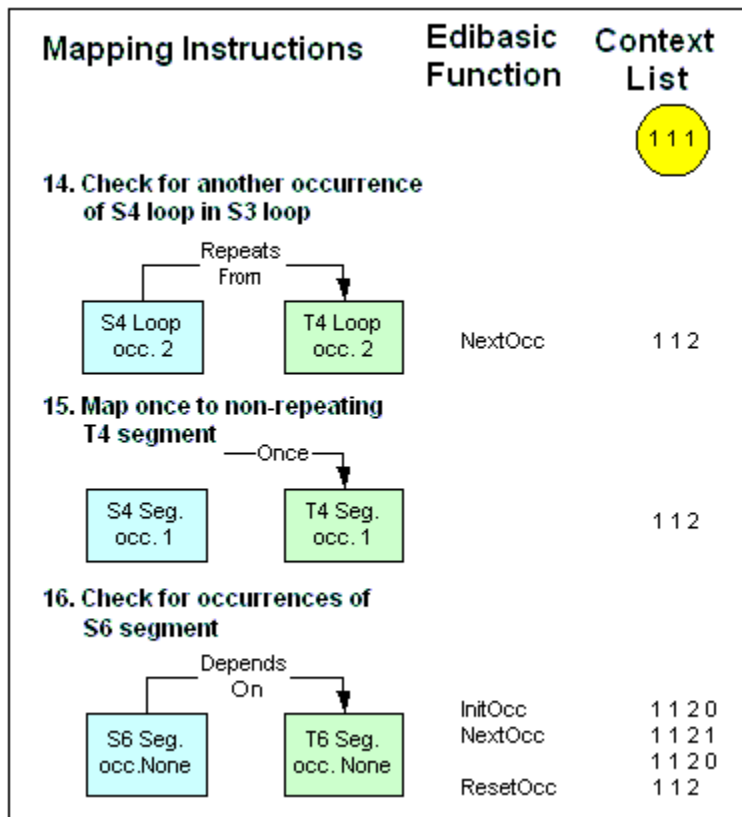
- After processing the first occurrence of the S5 loop, the context list is 1 1 1 1
- While processing occurrences of the S6 repeating segment, the context list is 1 1 1 1 1
- After finding no occurrences of the S6 segment, the context list reverts to 1 1 1 1
- After finding no more occurrences of the S5 loop, the context list reverts to 1 1 1



Effect of Mapping Equivalent Input and Output Structures on Context List, cont.

The next figure shows the effect of visual mapping on the second occurrence of the S4 loop. Not the following:

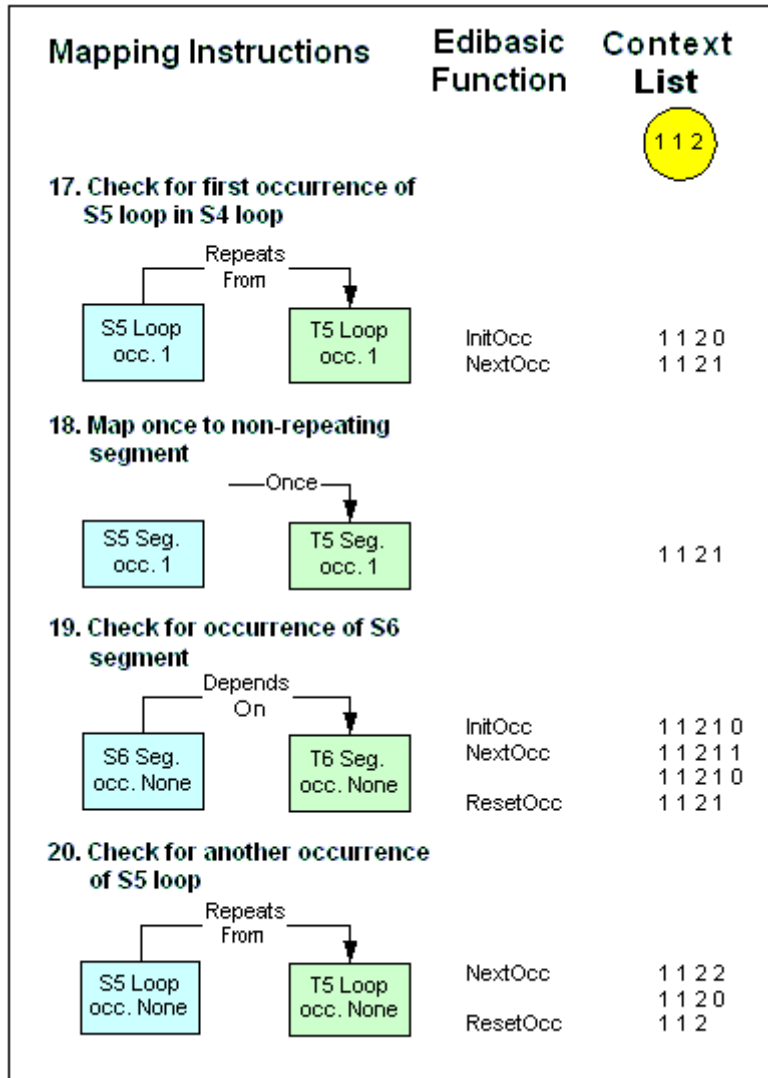
- After processing the second occurrence of the S4 loop, the context list is 1 1 2
- While processing occurrences of the S6 repeating segment, the context list is 1 1 2 1
- After finding no occurrences of the S6 segment, the context list reverts to 1 1 2



Effect of Mapping Equivalent Input and Output Structures on Context List, cont.

The next figure shows the effect of visual mapping on the first occurrence of the S5 loop within the second occurrence of the S4 loop. Note the following:

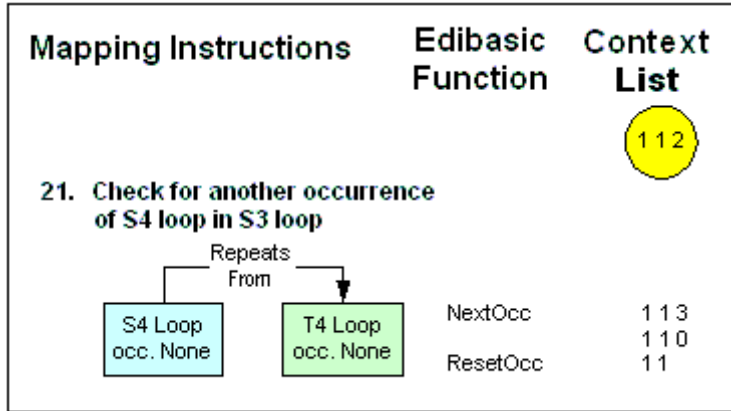
- After processing the next occurrence of the S5 loop, the context list is 1 1 2 1
- While processing occurrences of the S6 repeating segment, the context list is 1 1 2 1 1
- After finding no occurrences of the S6 segment, the context list reverts to 1 1 2 1
- After finding no more occurrences of the S5 loop, the context list reverts to 1 1 2



Effect of Mapping Equivalent Input and Output Structures on Context List, cont.

The next figure shows the effect of visual mapping when there are no more occurrences of the S4 loop. Note the following:

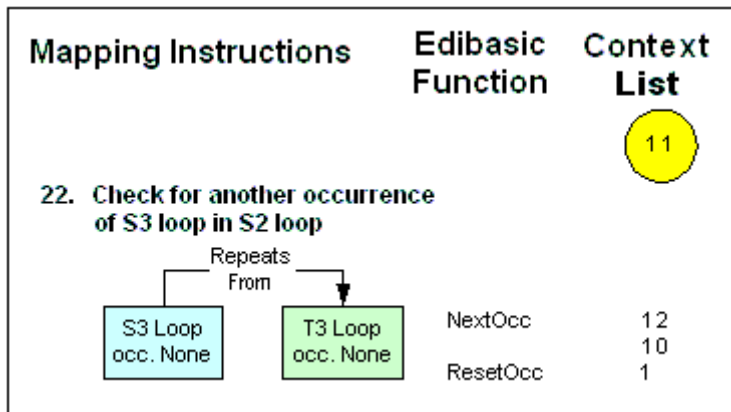
- After finding no more occurrences of the S4 loop, the context list reverts to 1 1



Effect of Mapping Equivalent Input and Output Structures on Context List, cont.

The next figure shows the effect of visual mapping when there are no more occurrences of the S3 loop. Note the following:

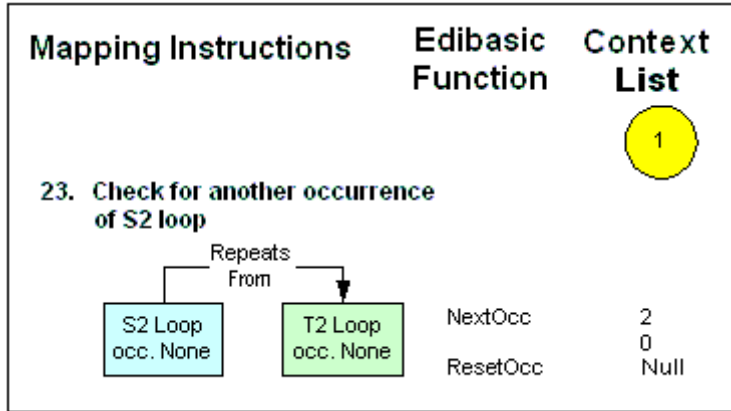
- After finding no more occurrences of the S5 loop, the context list reverts to 1



Effect of Mapping Equivalent Input and Output Structures on Context List, cont.

To complete the example, the next figure shows the effect of visual mapping when there are no more occurrences of the S2 loop. Note the following:

- After finding no more occurrences of the S2 loop, the context list reverts to Null



Effect of Mapping Equivalent Input and Output Structures on Context List, cont.


Effect of Edibasic Commands on the Context List

When you use visual mapping, the TRM maintains the context list using certain commands. Mappers may accomplish the same thing with Edibasic.

Each time the TRM accesses an entity deeper in the input document structure, it uses the existing values in the context list to create the list for the next level. To ensure the context list is correct for the next mapping instruction, it uses a series of commands that are also accessible through Edibasic:

InitOcc	<ul style="list-style-type: none"> ▪ sets the level to the level of the segment ▪ sets the occurrence to zero ▪ completes missing levels with a default value, 1
NextOcc	<ul style="list-style-type: none"> ▪ checks for the existence of another occurrence.
ResetOcc	<ul style="list-style-type: none"> ▪ reverts to the state of the context list before the previous InitOcc was issued.

IMPORTANT: Note that for **InitOcc**, when there is information missing for previous levels of the context list, a default value of 1 is inserted for the missing levels.

These commands may be entered on the **Start** or the **GetNext** tabs of the Edibasic Edit window. You may also enter the **ResetOcc** on the Stop page for a clearer view of the final state in which you are leaving the context list when it finishes executing your commands. You access the Edibasic Edit window by selecting the **Edibasic Method** button  from the Map window. The **GetNext** and **Stop** tabs are only available for destination entities that repeat, such as loops or segments.

The TRM uses the following principles to maintain the context list:

- The context list begins with a null value
- The context list builds on previous values
- When you do not map higher levels of loops, the TRM inserts a default value of 1 for the missing level
- Drag-and-drop from a loop invokes a Repeats From process, which performs the following tasks:
 - Initialize the next level of repetition (**InitOcc**)
 - Find and log the next occurrence (**NextOcc**), repeating until there are no more occurrences
 - Hold this value until all lower levels have been processed
 - When there are no more occurrences (**NextOcc**), reset the current level to zero occurrences
 - After mapping the entity, reset the context list (**ResetOcc**) to what it was before the previous initialization (**InitOcc**)
- Drag-and-drop from a repeating segment invokes a **Depends On** process, which performs the following tasks:
 - Initialize the next level of repetition (**InitOcc**)
 - Find and log the next occurrence (**NextOcc**)
 - After the last occurrence, reset the context list (**ResetOcc**) to what it was before the previous initialization (**InitOcc**)
- These commands do not modify the context list
 - The **Once** mapping command ignores the context list
 - Drag-and-drop from a non-repeating segment uses the current state of the context list

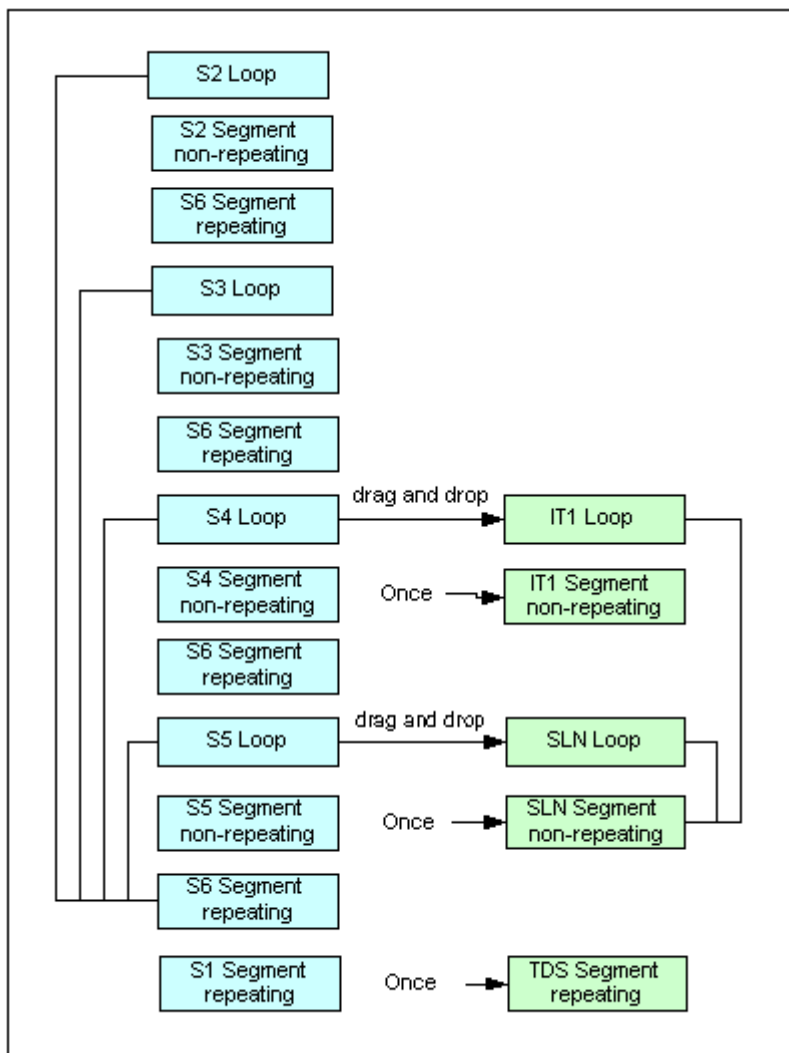
Example of Context List Using Drag-and-Drop for Dissimilar Structures

The following example shows how to use Edibasic commands to manipulate the context list when the output structure does not match the input structure.

Consider mapping the preceding input structure to a dissimilar output structure.

Let us consider one possibility, which is to drag and drop the S4 loop directly to the IT1 loop, without mapping any higher levels of the input loops or segments.

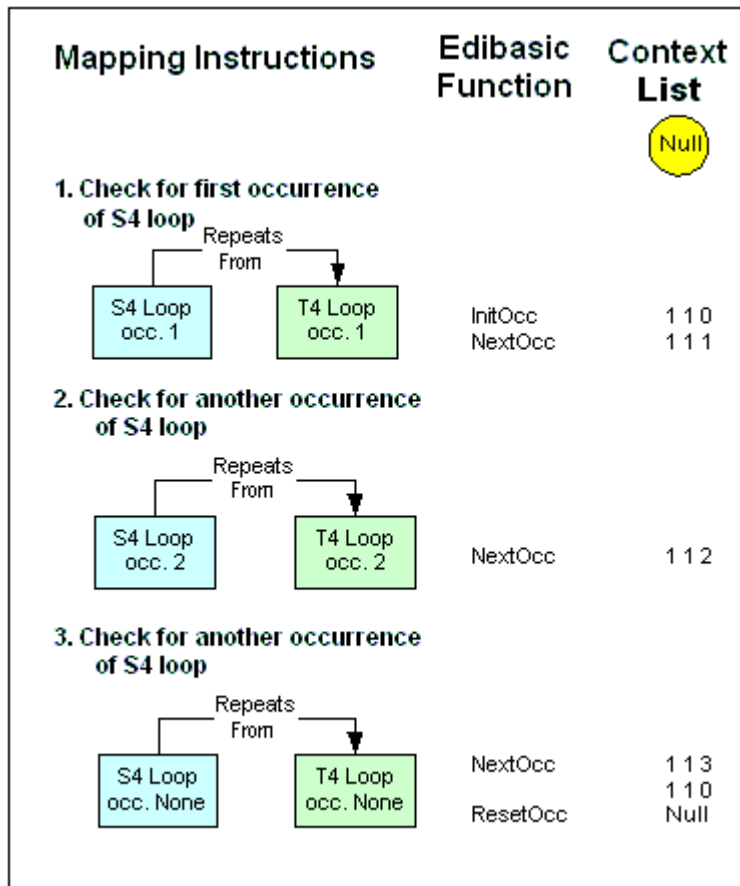
The following diagram is an overview of mapping the loop S4 directly to the loop IT1.



Overview of Mapping Techniques for Dissimilar Structures Using Drag-and-Drop

The next figure shows the effect of visual mapping on the first occurrence of the S4 loop, without mapping any higher levels first. Note the following:

- The context list begins with a null value
- The TRM inserts a default value of 1 for each missing level, e.g. for the S2 and S3 loops
- After the first occurrence of the S4 loop, the context list is 1 1 1
- After the second occurrence of the S4 loop, the context list is 1 1 2
- After finding no more occurrences of the S4 loop, the context list reverts to Null



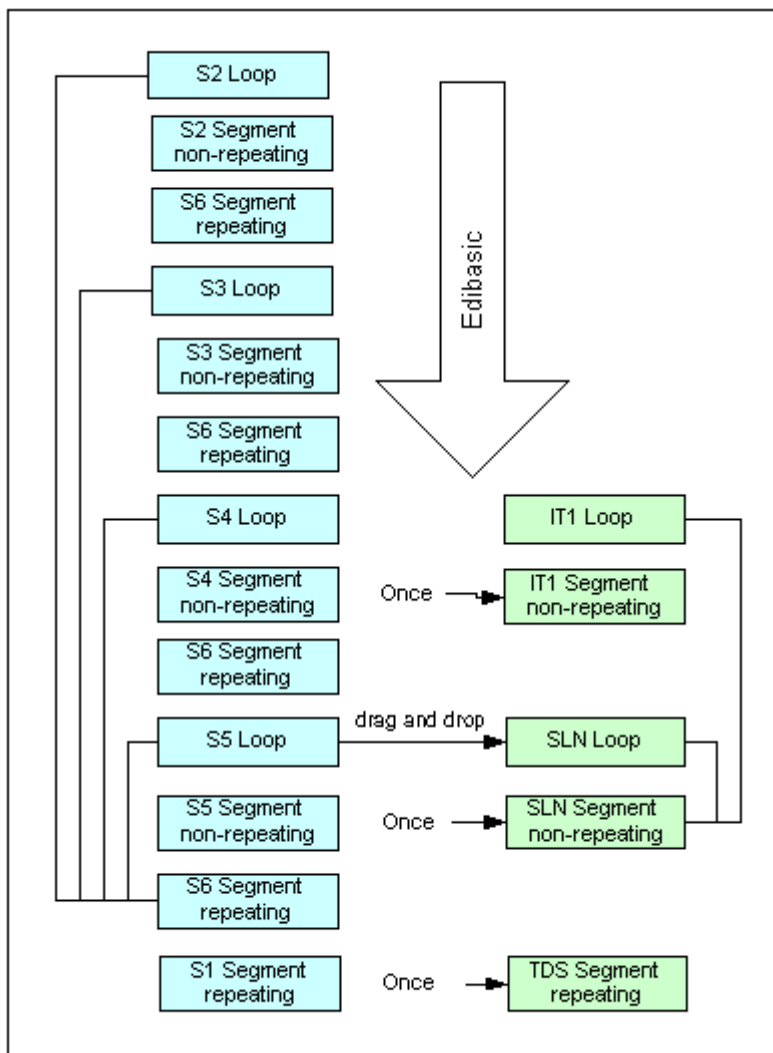
Effect of Mapping Dissimilar Input and Output Structures on Context List Using Drag-and-Drop

Using this method of mapping, the first 2 levels would always be 1, while the third level would increment with additional occurrences. There may be circumstances where this type of mapping would work for the type of input you receive. At the end, the context list will revert to a Null value, which is what it was before the previous **InitOcc**. This will work when you do not need to access any S4 data in other occurrences of the higher loops, S3 or S2.

Example of Context List Using Edibasic for Dissimilar Structures

The following diagram is an overview of mapping loop S2 through loop S5, this time using Edibasic to control levels and occurrences through the S4 loop.

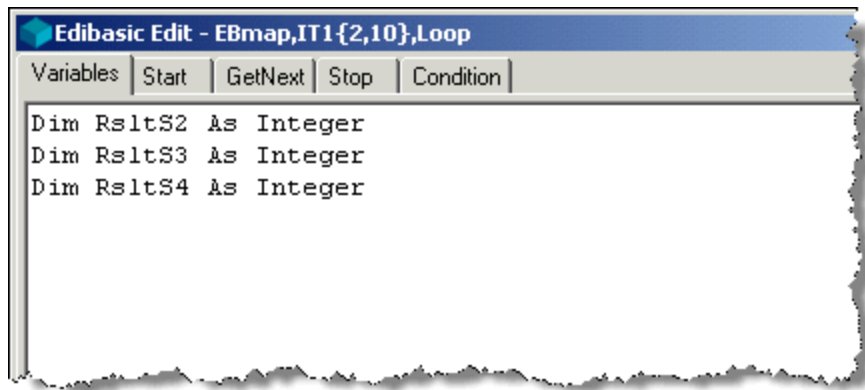
Again, the S1 segment as a global segment does not use the context list, so we begin with the S2 loop. We want to map element data to the IT1 only from the S4 loop and lower. We will use Edibasic commands to position the context list so it is pointing at the correct levels. Once we have done that, we may continue with drag-and-drop visual mapping. Notice that the last output segment, the TDS, is at the same level as the S1 input segment. Therefore, by the time we map the TDS segment, the context list must be null.



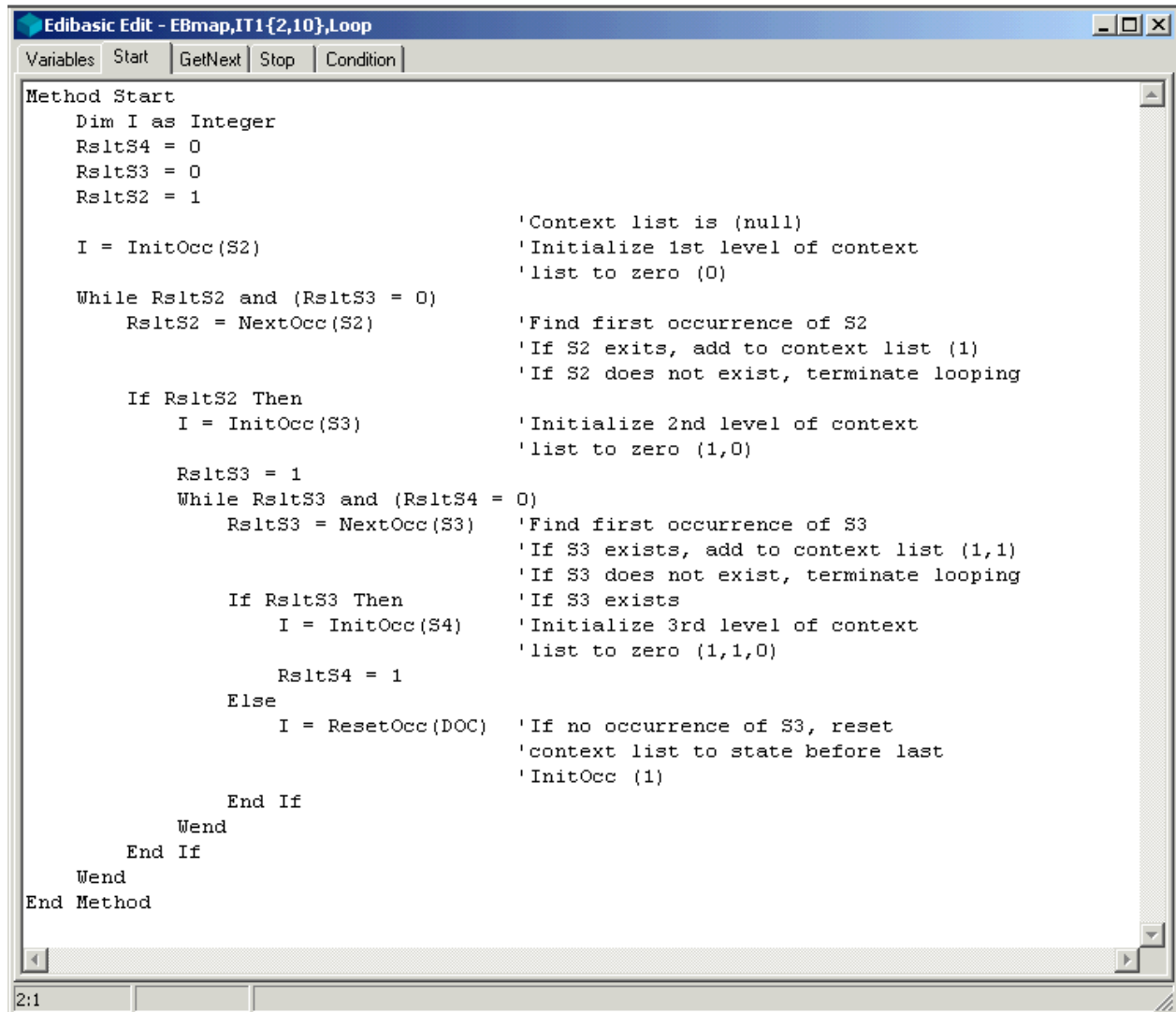
Overview of Mapping Techniques for Dissimilar Structures Using Edibasic and Drag-and-Drop

Here we want to allow for an unknown number of occurrences of the S2 and S3 loops. This example explains how to develop the context list so that you can allow for multiple occurrences of the S2, S3 and S4 loops and then use visual mapping at the lower levels.

NOTE: The Edibasic commands to manipulate the context list are placed on the output IT1 loop.

A screenshot of a software window titled "Edibasic Edit - EBmap,IT1{2,10},Loop". The window has a menu bar with "Variables", "Start", "GetNext", "Stop", and "Condition". The main area contains three lines of code: "Dim RsltS2 As Integer", "Dim RsltS3 As Integer", and "Dim RsltS4 As Integer".

```
Edibasic Edit - EBmap,IT1{2,10},Loop
Variables | Start | GetNext | Stop | Condition
Dim RsltS2 As Integer
Dim RsltS3 As Integer
Dim RsltS4 As Integer
```



The screenshot shows a window titled "Edibasic Edit - EBmap,IT1{2,10},Loop". The window has a menu bar with "Variables", "Start", "GetNext", "Stop", and "Condition". The main area contains the following BASIC code with comments:

```
Method Start
  Dim I as Integer
  RsltS4 = 0
  RsltS3 = 0
  RsltS2 = 1

  I = InitOcc(S2)           'Context list is (null)
                          'Initialize 1st level of context
                          'list to zero (0)

  While RsltS2 and (RsltS3 = 0)
    RsltS2 = NextOcc(S2)   'Find first occurrence of S2
                          'If S2 exists, add to context list (1)
                          'If S2 does not exist, terminate looping

    If RsltS2 Then
      I = InitOcc(S3)      'Initialize 2nd level of context
                          'list to zero (1,0)

      RsltS3 = 1
      While RsltS3 and (RsltS4 = 0)
        RsltS3 = NextOcc(S3) 'Find first occurrence of S3
                              'If S3 exists, add to context list (1,1)
                              'If S3 does not exist, terminate looping

        If RsltS3 Then
          I = InitOcc(S4)  'Initialize 3rd level of context
                              'list to zero (1,1,0)

          RsltS4 = 1
        Else
          I = ResetOcc(DOC) 'If no occurrence of S3, reset
                              'context list to state before last
                              'InitOcc (1)

        End If
      Wend
    End If
  Wend
End Method
```

The status bar at the bottom left shows "2:1".

The code on the Start page is executed once for all occurrences of the entity to which it applies, in this case the IT1 loop. The next figure shows the effect of Edibasic mapping on the first occurrences of the S2, S3 and S4 loops. Note the following:

- The Edibasic commands on the Start page on the IT1 loop set the context list by traveling down the hierarchy of the S2 loop, the S3 loop and to the S4 loop.
- The context list begins with a null value
- After the first occurrence of the S2 loop, the context list is 1
- After the first occurrence of the S3 loop, the context list is 1 1
- Before the first occurrence of the S4 loop, the context list is initialized to 1 1 0

Mapping Instructions	Edibasic Function	Context List
Start page of IT1 Loop		
		Null
1. Check for first occurrence of S2 loop		
Dim I as Integer RsltS4 = 0 RsltS3 = 0 RsltS2 = 1 I = InitOcc(S2)	InitOcc(S2)	0
While RsltS2 and (RsltS3 = 0) RsltS2 = NextOcc(S2)	NextOcc(S2)	1
2. Check for first occurrence of S3 loop		
I = InitOcc(S3)	InitOcc(S3)	1 0
RsltS3 = 1 While RsltS3 and (RsltS4 = 0) RsltS3 = NextOcc(S3)	NextOcc(S3)	1 1
3. Check for first occurrence of S4 loop		
If RsltS3 Then I = InitOcc(S4)	InitOcc(S4)	1 1 0

Effect of Mapping Dissimilar Input and Output Structures on Context List Using Edibasic

The following code appears on the **GetNext** tab. It loops through the input looking for occurrences of the S4 loop. The code on this page is executed to determine each occurrence of the output, which is based on testing the input. The visual mapping used to generate the element data and the lower level SLN loop depends on the context list with the requisite levels and the correct occurrences. Note the following:

- The **GetNext** function controls whether to generate occurrences
- The **NextOcc** function provides a return value that is held in the variables we created for each of the loop levels, RsltS2, RsltS3 and RsltS4
- The value in the variable is used to determine when to generate an occurrence using the **GetNext** function
- A zero value in the variable means that **GetNext** will also return a zero value, which does not generate an occurrence
- A non-zero value in the variable means that **GetNext** will return a value of 1, which does generate an occurrence

```

Method GetNext As Integer
  Dim I as Integer
  If RsltS3 = 0 Then
    GetNext = 0
  Else
    GetNext = 1
  End If
  RsltS4 = NextOcc(S4)

  While GetNext and (RsltS4 = 0)
    I = ResetOcc(DOC)

    RsltS3 = NextOcc(S3)

    While GetNext and (RsltS3 = 0)
      I = ResetOcc(DOC)

      RsltS2 = NextOcc(S2)

      If RsltS2 = 0 Then
        I = ResetOcc(DOC)

        GetNext = 0
      Else
        I = InitOcc(S3)

        RsltS3 = NextOcc(S3)

        End If
      End If
    End If
  End If
  I = InitOcc(S4)

  RsltS4 = NextOcc(S4)

  End If
Wend
End Method

```

'Holds return value of InitOcc, ResetOcc
 'If S3 does not exist,
 'Do not generate loop
 'If S3 exists,
 'Generate loop
 'Find next occurrence of S4
 'If S4 exists,
 'add to context list, (1,1,1)
 'If S3, but no S4,
 'reset context list to state
 'before last InitOcc, (1,1)
 'Find next occurrence of S3
 'If S3 exists,
 'add to context list, (1,2)
 'If another S3 does not exist
 'Reset context list to state
 'before last InitOcc, (1)
 'Find next occurrence of S2
 'If S2 exists,
 'add to context list, (2)
 'If another S2 does not exist
 'Reset context list to state
 'before last InitOcc (null)
 'Do not generate any more occurrences
 'If another occurrence of S2 exists,
 'Initialize 2nd level of context
 'list to zero, (2,0)
 'Find next occurrence of S3 in next S2
 'If S3 exists,
 'add to context list, (2,1)
 'If S3 exists,
 'Initialize 3rd level
 'of context list, (2,1,0)
 'Find next occurrence of S4 in next S3
 'If S4 exists,
 'add to context list, (2,1,1)

The next figure shows the effect of visual mapping on the first occurrences of the S4 loop and allows for other occurrences of the S4 loop in additional occurrences of the S2 and S3 loops. Note the following:

- The Edibasic commands on the **GetNext** tab on the IT1 loop set the context list by traveling through the hierarchy of the S4 loop within all possible additional occurrences of the S2 loop and the S3 loop
- When there are no more occurrences of the S4 loop, the Edibasic commands set the context list to look for additional occurrences of the S3 loop, 1 2, to then find additional occurrences of the S4 loop
- When there are no more occurrence of the S3 loop, Edibasic commands set the context list to look for additional occurrences of the S2 loop, 2, to then find additional occurrences of the S3 loop, 2 1, and then occurrences of the S4 loop, 2 1 1
- When there are no more occurrences of the S2 loop, the Edibasic commands reset the context list to what it was before it executed the commands on the **Start** tab, null, which is where it must be before you use drag-and-drop to map the TDS segment from the S1 segment

Mapping Instructions	Edibasic Function	Context List
GetNext page of IT1 Loop		Null
3. Check for first occurrence of S4 loop		
RsltS4 = NextOcc(S4)	NextOcc(S4)	1 1 1
4. When no more S4 loops, check for another occurrence of S3 loop and more S4 loops		
While GetNext and (RsltS4 = 0) I = ResetOcc(DOC)	ResetOcc(DOC)	1 1
RsltS3 = NextOcc(S3)	NextOcc(S3)	1 2
If GetNext Then I = InitOcc(S4)	InitOcc(S4)	1 2 0
RsltS4 = NextOcc(S4)	NextOcc(S4)	1 2 1
5. When no more S3 loops, check for another occurrence of S2 loop, another S3 loop and more S4 loops		
While GetNext and (RsltS4 = 0) I = ResetOcc(DOC)	ResetOcc(DOC)	1 2
While GetNext and (RsltS3 = 0) I = ResetOcc(DOC)	ResetOcc(DOC)	1
RsltS2 = NextOcc(S2)	NextOcc(S4)	2
6. Repeat step 4		
7. When no more S2 loops, initialize context list to null		
If RsltS2 = 0 Then I = ResetOcc(DOC)	ResetOcc(DOC)	Null

Accessing Input Directly Using Edibasic without Using the Context List

There may be times that you want to access input data that is outside of the current location of the context list. You can do this using Edibasic and directly referencing the input item without using **InitOcc**, **NextOcc** and **ResetOcc** functions. When you reference an element directly, the TRM uses a separate, temporary list that does not affect the context list. You must, however, use the same level designations, e.g. the same vectors, specified in the context list to access that data.

IMPORTANT: Remember that the level given in the document definition does not always match the actual level required to track occurrences for mapping. The TRM assigns an additional level to each repeating segment and loop during processing. In the preceding table, the access level for loop S2 is 1, although the document definition level is 0. Similarly, the access level to map repeating segment S6 is 2, even though the document definition level is 1. Therefore, to map from a loop or repeating segment, mappers must always add 1 to the level listed in the document definition for that loop or repeating segment.

Edibasic Solutions

Here are some solutions to basic problems using Edibasic.

Using Conditions to Create Output

You can write conditions that apply to certain entities within an output document, affecting how and when the output is generated. You write conditions within the **Method Condition** on the **Condition** tab of the Edibasic Edit window for the entity to which it applies.

Within mapping, you have several techniques that could potentially generate (map) an entity. The following table specifies what technique does what and when, assuming there is no **Method Condition** attached to the entity.

Mapping Technique	Result
Drag-and-drop	Produces output only if there is input data. For loops, if the first segment of the loop contains no data, the loop is not created, even though segments within the loop may have data.
Literal	Always produces output.
Internal Field	Produces output if the value in the field is not null.
Edibasic: GetNext	Produces output if the value returned is not zero.
Edibasic: MapEle	Produces output if the value returned is not null.

However, **Method Condition** always has the last say. You can write condition instructions for both visual mapping and Edibasic techniques, but it must be associated with a mapping technique. A Method Condition alone cannot cause an entity to be mapped.

The following table describes how **Method Condition** affects the other mapping techniques.

Return Code Value	Result
Zero	Output is not created, which overrides the mapping technique associated with the entity.
Non-zero	Output is created as indicated by the associated mapping technique.

IMPORTANT: If you enter any Edibasic instructions for a **Method Condition**, it always returns a value. If you have not specified a return value, the default value is zero, which does not create the entity. Therefore, when you enter code on a **Condition** tab, you should always initialize **Condition** to **1** at some point to make sure you create output.

Mapping Values from an Input Trailer Wrapper

MW Translator parses data and places it in the Data Element Store (DES) in sequence as it parses the input. Because of this sequence, not all data is available in the DES during mapping.

For example, data from the input wrapper trailer is not available until after all the documents have been parsed and mapped, so you would not be able to access the data from a wrapper trailer in a document map.

Also, you cannot access the information for an input wrapper trailer from a wrapper map. You can access the information just after it has been parsed and pass that value to the wrapper map using a global variable. Here are the basic steps:

- On the definition for the input wrapper, declare a global variable. For instructions, refer to the topics:
 - *To Declare Variables for Validate Routines on Wrappers* (on page 672)
 - *To Declare Variables for Validate Routines on Segments in Wrappers* (on page 673)
- On the definition for the input wrapper, place the code that stores the value in the global variable on the **Validate** tab for a loop, segment, composite or element.
- In your wrapper map, again declare the global variable to set the scope so that MW Translator can access the data.
- In your wrapper map, enter the code that accesses the information in the global variable.

Using Edibasic to Specify Output Locations

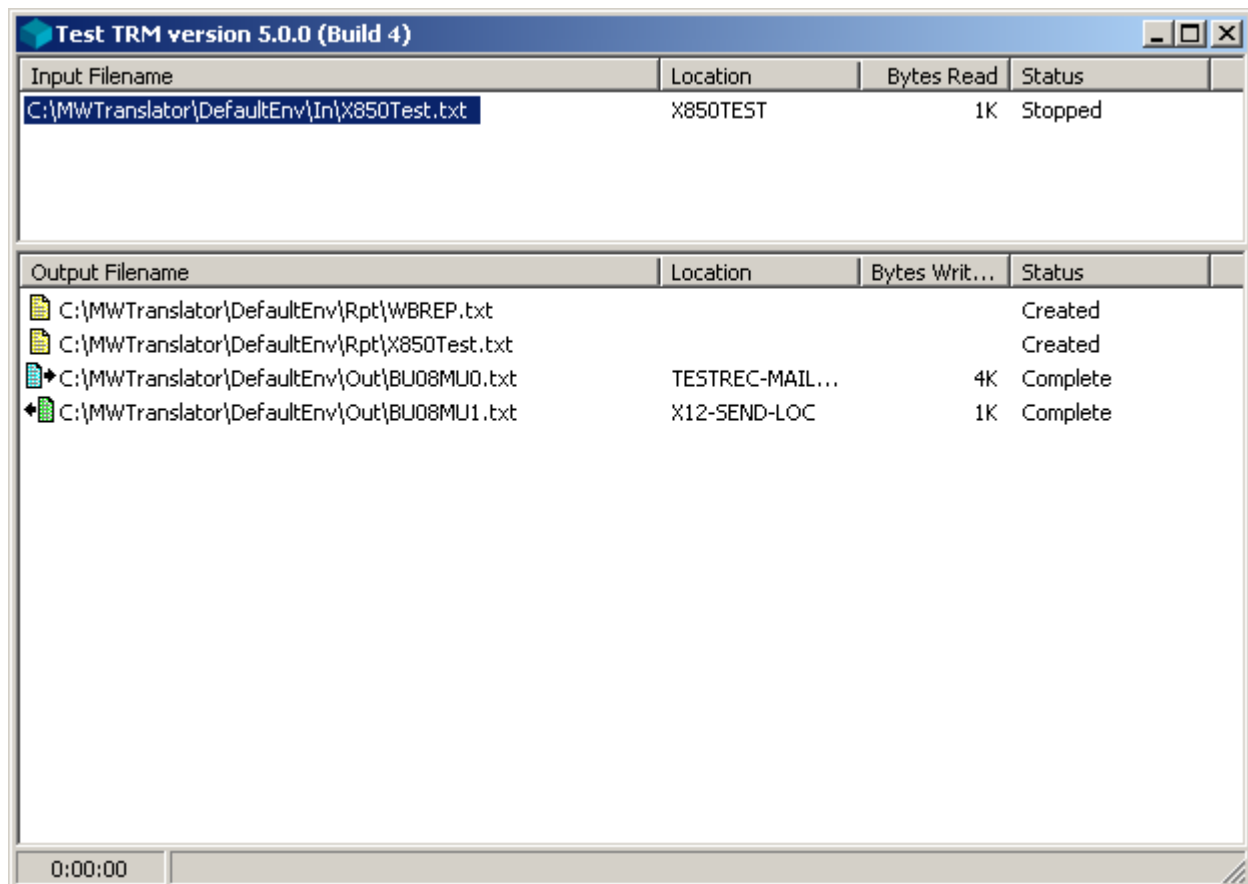
You can use Edibasic to specify the source or destination locations for the output for a particular map. The following example shows you how to modify the map used for the Example to do this. You will modify the Example-X850 map at the document level to:

- Create a dummy integer variable on the **Variables** tab to receive the return value of the **SetField** function, and
- Add Edibasic instructions on the **Condition** tab using the **SetField** function to map location values to the appropriate internal fields for the source and destination locations.

To Run an Initial Test

Make sure the Example definitions are already in your database. To quickly check this, look for the map, Example-X850, in the **Maps** folder. If it is not there, import the file **WORKBENCH\EXAMPLES\X850TEST\X850TEST.TRN**. For specific instructions, refer to the **readme** file in the same subdirectory. Run the test as specified in this file, or follow the instructions in the section, *Testing Configurations* (on page 387).

After you have run your test, notice that the destination location for the output is TESTREC-MAILBOX.




The screenshot shows a window titled "Test TRM version 5.0.0 (Build 4)". It contains two tables: "Input Filename" and "Output Filename".

Input Filename	Location	Bytes Read	Status
C:\MWTranslator\DefaultEnv\In\X850Test.txt	X850TEST	1K	Stopped

Output Filename	Location	Bytes Writ...	Status
C:\MWTranslator\DefaultEnv\Rpt\WBREP.txt			Created
C:\MWTranslator\DefaultEnv\Rpt\X850Test.txt			Created
C:\MWTranslator\DefaultEnv\Out\BU08MU0.txt	TESTREC-MAIL...	4K	Complete
C:\MWTranslator\DefaultEnv\Out\BU08MU1.txt	X12-SEND-LOC	1K	Complete

At the bottom left of the window, a timer displays "0:00:00".

You can open the processing report by selecting the report icon, . On the report will see that the source and destination locations are listed as follows:

Output number: 1 Source: TESTSEND-MAILBOX Destination: TESTREC-MAILBOX

To Create the Integer Variable

You use the **SetField** function to map values to some internal fields that will be used by the TRM during generation or routing. You can find a list of possible functions by searching for **SetField** in the online help. In this case, we will map values to the internal fields for the source location, F_SEND_MAILBOX, and the destination location, F_REC_MAILBOX.

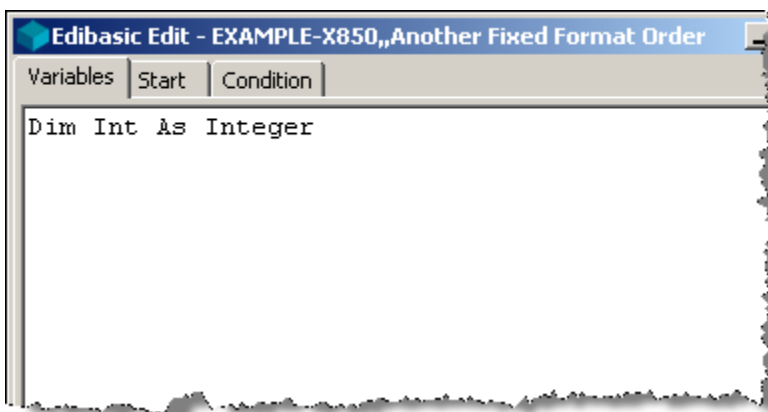
NOTE: Since the **SetField** function returns an integer value, you must create a dummy integer variable to satisfy syntax requirements. If you do not create an integer variable, the TRM would create a temporary variable as a variant, which will not satisfy the syntax requirement. When you check your syntax or generate the map an error will be displayed.

- 1 From Data Explorer, open the Example-X850 map.
The Map window appears.
- 2 From the header area of the destination document, select the **Document Variables and Methods** button



The Edibasic Edit window appears.

- 3 On the **Variables** tab, enter the following Edibasic code to create the variable.



Remember that variable names are not case sensitive. You created a local variable using the **Dim** statement rather than a global variable using the **Global** statement, because you do not need this variable to exist beyond the lifetime of the document or interchange.

- 4 **Check your syntax for errors** (on page 585). Do not close this window.

To Map Locations for the Output

Make sure you have completed the *previous instructions* (on page 215).

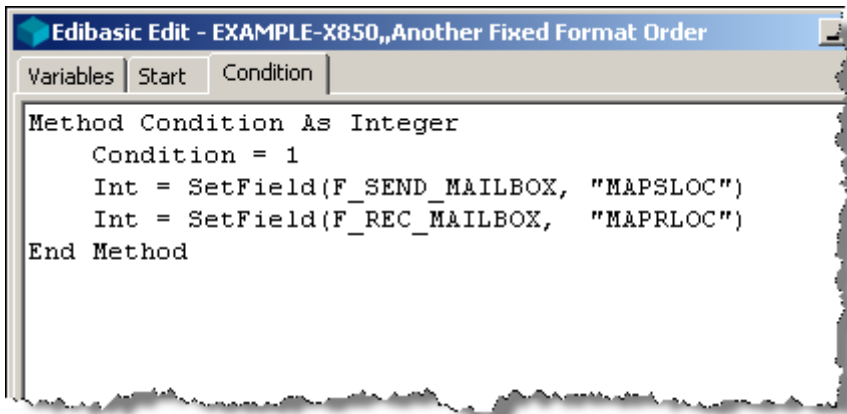
- 1 Select the **Condition** tab.

A dialog box appears asking you if you want to save your changes.

- 2 Select **Yes** to save your changes.

The Condition tab appears.

- 3 On the **Condition** tab, enter the following Edibasic code:



```

Edibasic Edit - EXAMPLE-X850,,Another Fixed Format Order
Variables Start Condition
Method Condition As Integer
  Condition = 1
  Int = SetField(F_SEND_MAILBOX, "MAPSLOC")
  Int = SetField(F_REC_MAILBOX, "MAPRLOC")
End Method

```


Notice that you first set the condition to **1**, which forces it to create output. If you do not do this, the condition is initialized to zero as a default, which creates no output. Then you use **SetField** to store your location values in the internal fields for the output source and destination locations.

- 4 *Check your syntax for errors* (on page 585).
- 5 Close this window, saving your changes.
- 6 Close the Map window.

To Test Your Mapping Instructions

- 1 From the **Generate** menu, select **All**.

Fix any errors related to your test before you proceed.

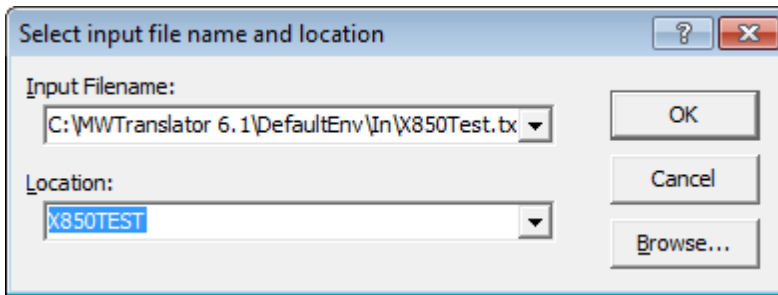
- 2 From the menu bar, select the **View Test Setup** button .

The Test window appears.

- 3 Select the **Add Test Input** button .

The **Select input file name and location** dialog box appears.

- 4 Select values as follows (your directory may be different):

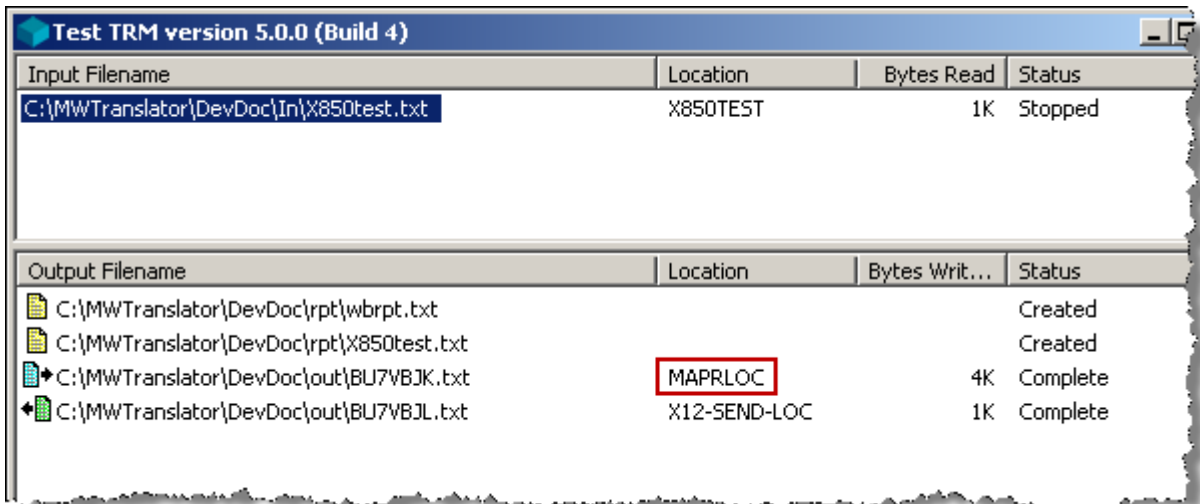


- 5 Select **OK**.


The Test window reappears.

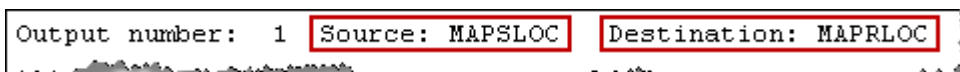
- 6 Select the **Run Test** button.

Your Test window should appear similar to the following:



Note that the destination location for the output is now **MAPRLOC**.

You can open the processing report by selecting the report icon, . On the report will see that the source and destination locations are listed as follows:



This page intentionally blank

Defining Trade Agreements and Acknowledgments

Overview

The trade agreement profile provides mandatory information for processing inbound data. If a matching trade agreement is not found, the TRM terminates processing. It packages existing definitions so they can be called together as needed. This avoids making you specify the pieces every time you want to reuse the same definitions.

The acknowledgment profile provides mandatory information for generating acknowledgments. Unlike trade agreements, however, if a matching acknowledgment is not found, processing continues anyway. The acknowledgment profile also packages existing definitions so they can be called together as needed.

Here you learn two basic tasks:

- How to define trade agreement profiles
- How to define acknowledgment profiles

Additionally, there is a description of how to define a basic proprietary acknowledgment, describing some of the issues and constraints. This should give you enough information to create and test one of your own.

Trade Agreement Profiles Process Inbound Data

The TRM must find a trade agreement profile or processing terminates. To find the trade agreement profile, it will try to match one from a list of trade agreement profiles associated with any of these definitions, in the following order, from most restrictive to least restrictive:

- Partner Relationship
- Closed Group (Recipient Partner or Standard ID in a Group)
- Recipient Partner
- Standard ID

Which option you choose depends on your trading requirements. For more information about these options and their implications, refer to the topic, "Specifying a Trade Agreement Profile" in the "Determining Processing" chapter.

NOTE: When you receive acknowledgments, which are a type of inbound document, you must create a trade agreement profile to do the processing, just as you would for any inbound document. For an example of one way to do this, refer to the topic, *Configuring a Trade Agreement to Process an Acknowledgment without Mapping* (on page 239).

Trade agreement profiles contain the following information for processing, some of which are specified on an as-needed basis:

- Input standard ID and other values from wrapper for matching (input wrapper was already found during standard identification and used to parse the wrapper data)
- Input document, security document, security user exit
- Action, error action
- List of outputs
- Output properties: output document, wrapper, document map, wrapper map, summary document map, security user exit, alias type

Acknowledgment Profiles Generate Outbound Acknowledgments

The TRM will generate an acknowledgment if one matches the wrapper of the inbound data. To find the acknowledgment profile, it will try to match one from a list of acknowledgment profiles associated with any of these definitions, in the following order, from most restrictive to least restrictive:

- Partner Relationship
- Closed Group (Sending partner or Standard ID in a Group)
- Sending Partner
- Standard ID

Which option you choose depends on your trading requirements. It is likely that you will have only one or a very limited number of acknowledgments for a given standard. Therefore, it is likely that you would associate it with the least restrictive definition, the Standard ID (inbound wrapper standard)

Acknowledgment profiles contain the following information for processing:

- Input wrapper
- Acknowledgment map, wrapper map
- When to create acknowledgment, when to report document status
- summary document map, security user exit

Note that aliasing is not used for acknowledgments.

If you are not using one of the defined acknowledgment definitions, that is the X12 997 or the EDIFACT CONTRL, you must define one before you can use it in an acknowledgment profile. Any acknowledgments you define are called proprietary acknowledgments.

There is a related feature in MW Translator that allows you to track outbound documents with responding acknowledgments. This process is called reconciliation. Reconciliation relies on collecting data, a process called audit. Audit and reconciliation are controlled from the Operator program. For more information, refer to the chapter "Using Audit and Reconciliation" in the *MW Translator Operator Program Guide and Reference* and in the online help for the MW Translator Operator Program.

Task 3. Defining Trade Agreement Profiles

To add a trade agreement profile, you select the **Trade Agreements** folder from Data Explorer, and then **Add** from the **File** menu.

After the TRM has selected the wrapper definition and parsed the incoming wrapper, it must find the trade agreement profile so it knows what else it must do. If the TRM cannot find the trade agreement profile that matches its criteria, it terminates processing.

You can later add this profile to a list of trade agreements for the recipient partner, partner relationship or standard ID.

Task 3.1. Specifying the Matching Fields

When it looks for a trade agreement profile to determine what it must do, the TRM must use some criteria to find it. Those criteria are the values in the fields on the **Matching Fields** tab of the Trade Agreement Profile window.

First the TRM must match the standard wrapper selected for the inbound standard with the value in the **Identified Standard** field. Then it will attempt to match the remaining information. For a detailed description of how it matches this data, refer to the chapter, *Understanding Processing Flow* (on page 45).

The screenshot shows a dialog box titled "Trade Agreement Profile: EXAMPLE-X850" with a "Matching Fields" tab selected. The dialog is divided into two main sections: "Identified Standard:" and "Input Field Values:".

Identified Standard:	Input Field Values:
X12	Doc Id: 850
	Version: 003030
	Release:
	Agency: X
	Assoc:

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

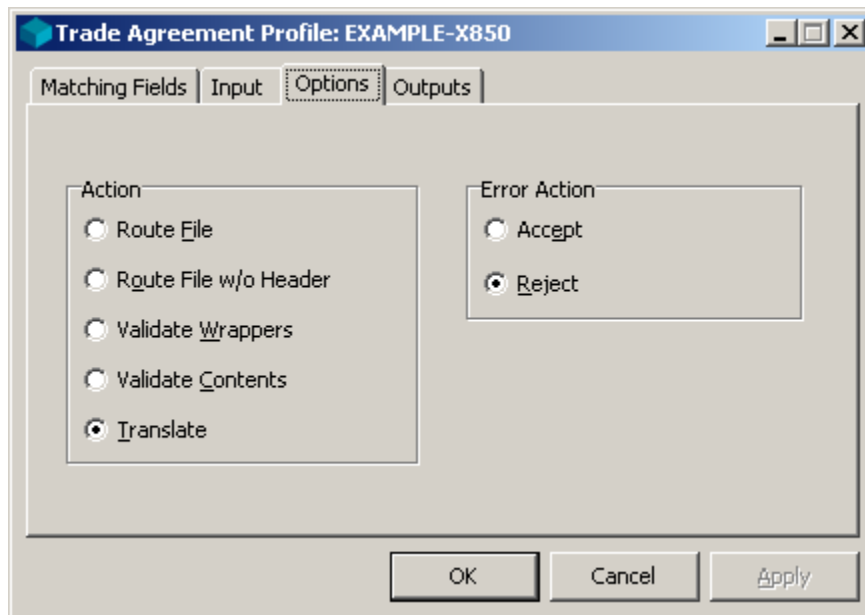
Task 3.2. Specifying the Options

Once it has selected a particular trade agreement profile, the TRM will perform the work you specified on the **Options** tab. You may choose among several TRM tasks:

Action	Description
Route File	Route a file whose content structure may not be defined. The input file may or may not have a header. Validate the header when a definition exists and route. Not valid for XML input.
Route File w/o Header	Route a file whose content structure may not be defined. The input file must have a header that is defined, which will be validated and then removed before routing. Not valid for XML input.
Validate Wrappers	Route a file whose wrapper structure is defined. Check wrapper data for compliance, then route the data without modification. Not valid for XML input.
Validate Contents	Route a file whose wrapper and content structure are defined. Check wrapper and contents for compliance, then route the data without modification. Not valid for XML input.
Translate	Translation implies validation of the wrappers and contents. In addition, this action performs translation(s) of the document based on the information in the Trade Agreement Output Properties associated with this trade agreement.

IMPORTANT: When you create a trade agreement profile for an inbound acknowledgment that you want to use for reconciliation, make sure that you select **Accept** as the error action. If there is an error on the acknowledgment, it will have a status of **E**, accept with errors, but it will still be used for reconciliation. If the acknowledgment is rejected, it cannot be used for reconciliation. For instructions to create a trade agreement profile for an inbound acknowledgment, refer to *Configuring a Trade Agreement to Process an Acknowledgment without mapping* (on page 239).

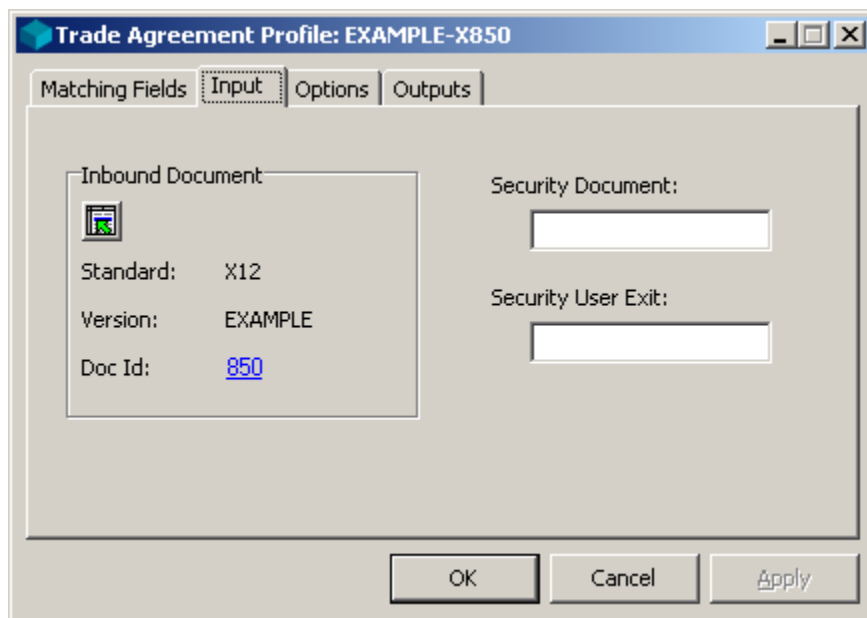
If you only need to validate wrappers and then route the data, you do not have to identify anything else. If you must validate contents or translate, then you must also specify the input document so it can parse the data. If you are translating, you must specify all of the outputs that it will create



Task 3.3. Specifying the Input Document

If you are validating the contents or translating, you must identify the structure of the input document used to parse the data.

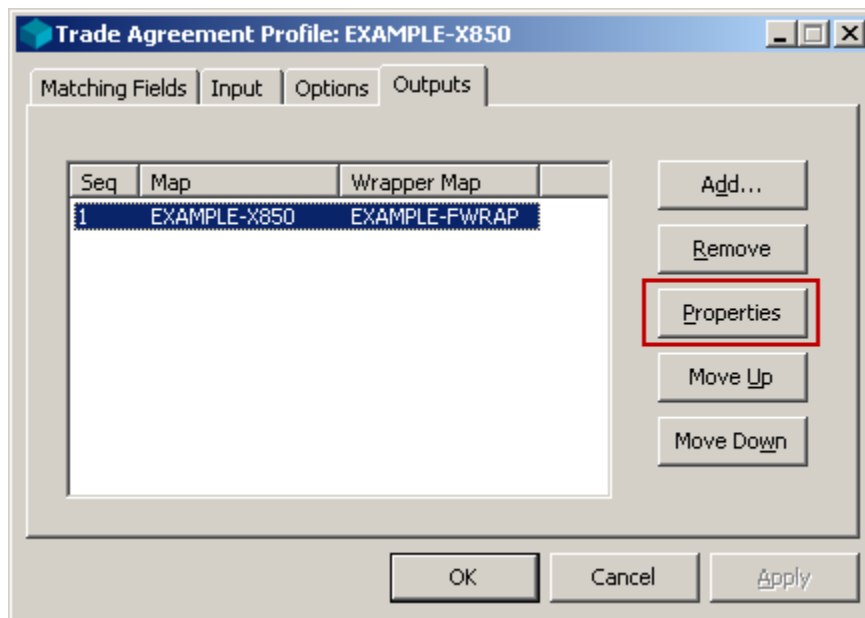
You can also specify a security document and a security user exit here for the input stream. You can configure a security user exit for an input stream (in the trade agreement profile) or an output stream (in the trade agreement profile or acknowledgment profile) and it will be called for each segment of that stream. The security user exit is useful, for example, for generating hash totals for parts of an input or output stream.



Task 3.4. Specifying the Outputs

If you are translating, you can create multiple outputs. This allows you to use different maps to process the same input. You specify all outputs for a single input document on the **Outputs** tab.

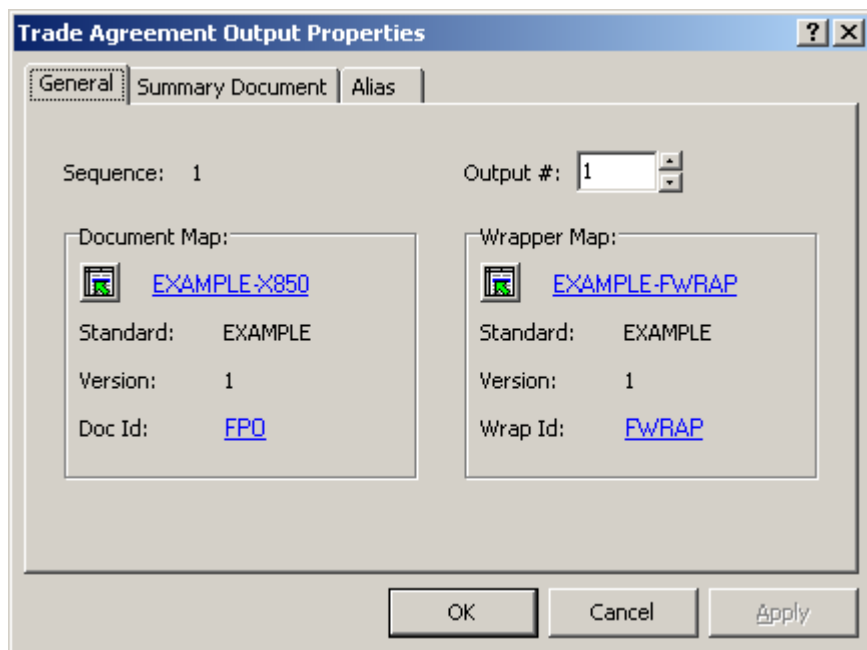
When you have selected multiple outputs, you can change the order in which the outputs are processed by using the **Move Up** and **Move Down** buttons. You then define the maps used to create the output wrapper and document in the output properties.



Task 3.5. Defining the Trade Agreement Output Properties

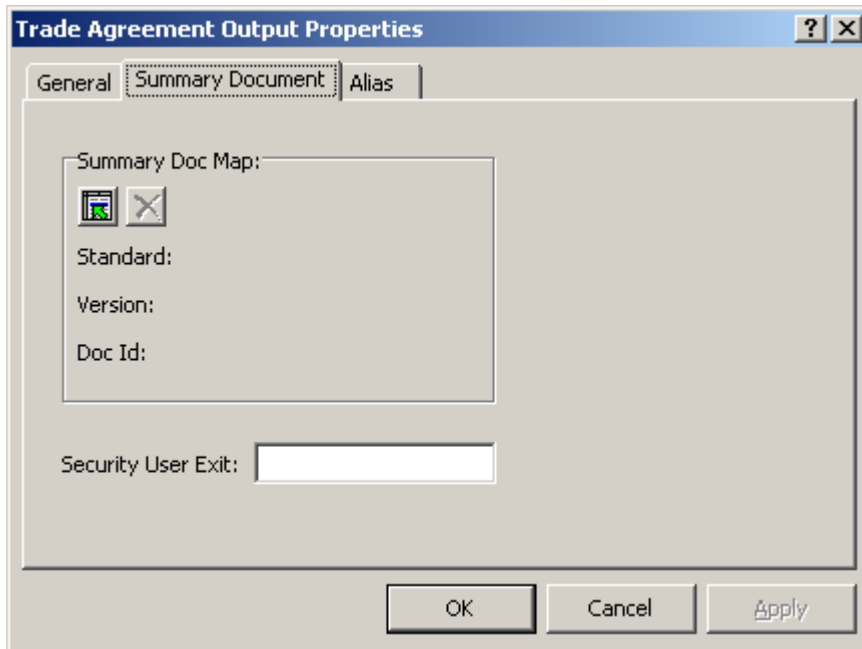
For each output you must define its output properties: the output document, the output wrapper, and the wrapper and document maps.

You can change the order in which the outputs are grouped, by changing the output number. For more information about how the TRM groups documents for output, refer to the topic, *Step 14. Generate (Validation Only or Translation)* (on page 86) in the chapter, *Understanding Processing Flow* (on page 45).



You can also specify a summary document and a security user exit that might be used to generate the data for the summary document on the Summary Document page.

You define a summary map to be used to generate a summary document at the end of an interchange. The EDIFACT AUTACK document is an example of a summary document. For more information, review the AUTACK example and its README file in the **Examples** subdirectory.



The **Alias** page allows you to specify an alias type for this output. This is one of the steps required to configure aliased partners so you can use alternative IDs in output wrappers and locations.

Aliases are one type of linking. There are 2 types of linking of partner records that allow you to specify alternative output IDs and locations, both of which are discussed in the chapter, *Defining Partners* (on page 243).




Task 3.6. Generating the Trade Agreement Profile

You must also generate a text file for the trade agreement and acknowledgment profiles you create. This one file contains all definitions, so you only need to do it once after you have created all trade agreement and acknowledgment profiles and before you test.

To generate this file select **Profile File** from the **Generate** menu.

Task 3.7. Printing a Trade Agreement Profile Report

To view a trade agreement profile report, select the trade agreement in the right pane of Data Explorer. From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file
- The **Print Preview** button  to print reports to the screen
- The **Print to PDF** button  to print to a PDF file

MW Translator Workbench Trade Agreement Profile Report

ProfileName:EXAMPLE-X850

Identified Standard: X12		
Input Field Values:	Inbound Document:	Action (invalid)
Doc Id 850	Standard X12	
Version 003030	Version EXAMPLE	Error Action (invalid)
Release	Doc Id 850	
Agency X		
Assoc		
Security Document	Security User Exit	
Outputs		
Seq 1	Output # 1	Alias Type
Document Map EXAMPLE-X850	Wrapper Map EXAMPLE-FWRAP	Summary Map
Standard EXAMPLE	Standard EXAMPLE	Standard
Version 1	Version 1	Version
Set ID FPO	Wrap ID FWRAP	Wrap ID
Security User Exit		

Task 4. Defining Acknowledgment Profiles


Typically, the only thing you need to do to use acknowledgments is to define an acknowledgment profile. The acknowledgment definitions and maps for X12 and EDIFACT standards are predefined and loaded with the standard versions. You can also define a proprietary acknowledgment, which is discussed in the topic, *Understanding Acknowledgments* (on page 235).

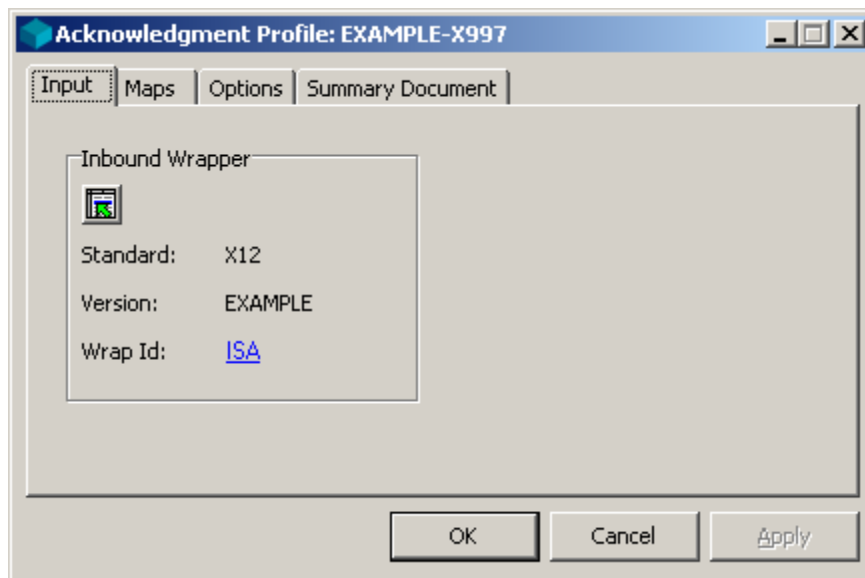
There are potentially four pages of information on the Acknowledgment Profile window.

Once the acknowledgment profile has been defined, you can add it to an acknowledgment profile list associated with a sending partner, partner relationship, or standard ID for default processing.


Task 4.1. Specifying the Input Wrapper

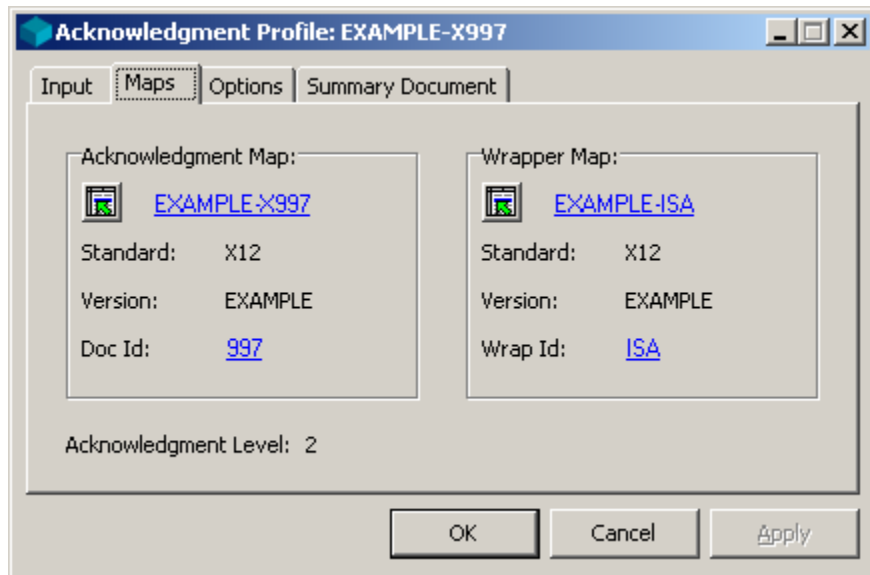
To find an acknowledgment profile, the TRM matches the input wrapper found during the standard identification process with the inbound wrapper identified on the **Input** page of the Acknowledgment Profile window.

You can choose the inbound wrapper definition for it to match using the selection button .



Task 4.2. Specifying the Maps

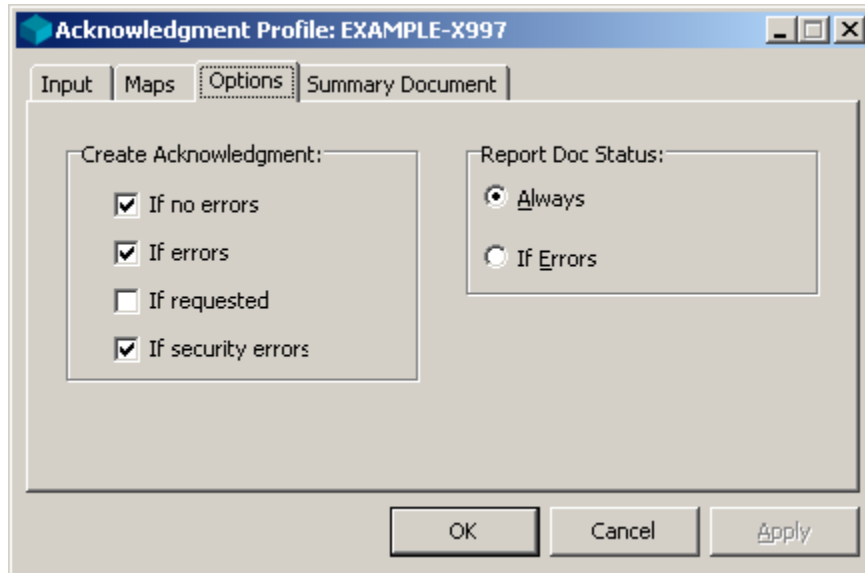
You must specify what maps to use to generate the acknowledgment and its wrapper. You choose the maps using the selection buttons .



Task 4.3. Specifying the Options

Once it has selected a particular acknowledgment profile, the TRM will perform the work specified on the **Options** tab.

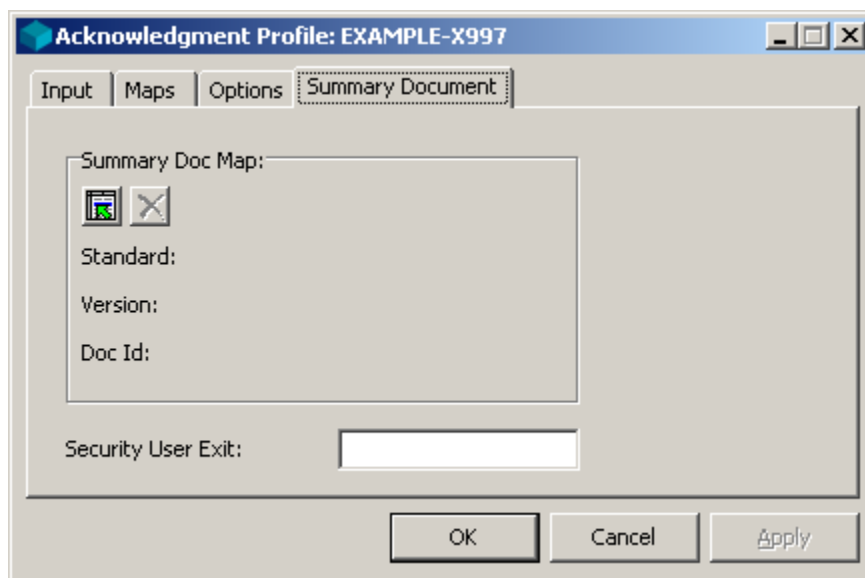
You can indicate when it should create an acknowledgment and whether it should report document status information.



Task 4.4. Specifying the Summary Document and Security User Exit

You can also specify a summary document and a security user exit that might be used to generate the data for the summary document on the **Summary Document** tab.

You define a summary map to be used to generate a summary document at the end of an interchange. The EDIFACT AUTACK document is an example of a summary document. For more information, review the AUTACK example and its **readme** file in the **Examples** subdirectory. Also, refer to the topics beginning with *User Exits Overview* (on page 329) in the chapter *Validation Routines and User Exits* (on page 323).






Task 4.5. Generating the Acknowledgment Profile

You must also generate a text file for the trade agreement and acknowledgment profiles you create. This one file contains all definitions, so you only need to do it once after you have created all trade agreement and acknowledgment profiles and before you test.

To generate this file select **Profile File** from the **Generate** menu.

Task 4.6. Printing an Acknowledgment Profile Report

To produce an acknowledgment profile report, select the trade agreement from the right pane of Data Explorer. From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file
- The **Print Preview** button  to print reports to the screen
- The **Print to PDF** button  to print to a PDF file

MW Translator Workbench Acknowledgment Profile Report

Profile Name:EXAMPLE-X997

<p>Input wrapper</p> <p>Standard X12</p> <p>Version EXAMPLE</p> <p>Set ID ISA</p>	<p>Create Acknowledgements</p> <p>If No Errors True</p> <p>If Errors True</p> <p>If Security Errors True</p> <p>If Requested False</p>	<p>Level 2</p> <p>Report Doc Status True</p>
<p>Document Map EXAMPLE-X997</p> <p>Standard EDIFACT</p> <p>Version 4</p> <p>Set ID AUTACK</p>	<p>Wrapper Map EXAMPLE-ISA</p> <p>Standard EDIFACT</p> <p>Version 4</p> <p>Wrap ID AUTACK</p>	<p>Summary Map</p> <p>Standard</p> <p>Version</p> <p>Wrap ID</p>

Security User Exit

Understanding Acknowledgments

When an application sends business information to another application, it might expect an acknowledgment responding to the information it sent. The application that received the business information would then create an acknowledgment to be returned to the sender of the original message. The acknowledgment indicates the acceptability of the original information.

Acknowledgments are often defined as part of a public standard, but they can be defined for any standard. Acknowledgments serve different purposes and may respond to different levels of information, depending on the standard. An acknowledgment conveys its message using statuses. MW Translator directly supports acknowledgments for X12 and EDIFACT, because it supplies all necessary configurations and definitions. MW Translator indirectly supports other types of acknowledgments, called proprietary acknowledgments. Users must supply definitions and configurations for any proprietary acknowledgment.

You might create a proprietary acknowledgment to send information back to an internal application. You might also create a proprietary acknowledgment to meet the requirements of a standard that MW Translator does not directly support. If you want to exchange a proprietary acknowledgment with a trading partner, that partner's system should be able to process the acknowledgment, assuming the partner wants to extract information from the acknowledgment.

Methods to Report Acknowledgment Information

You can report acknowledgment information in different ways. The most typical way is to use the capabilities of the TRM to automatically generate the acknowledgment information. This is the same process used to generate the public standard acknowledgment, the X12 997 and the EDIFACT CONTRL. The TRM gathers information in internal tables as it processes the inbound, subject document. It then generates an acknowledgment based on configurations and definitions. The TRM manages and reports all statuses and automatically configures routing. You can generate this type of acknowledgment at any of the wrapper levels as well as the document level.

Advanced users who find the typical type of acknowledgment process does not satisfy their needs may try another solution. They may create an acknowledgment as an outbound document whose information and statuses are mapped to the document. The user must manage statuses and any routing back to the sender. This method would require that users specify two outbound documents in the trade agreement profile to be processed in the following order: the acknowledgment and the original output. For example, if a user wants to reject a document based on analysis of inbound data, he would do so using Edibasic validation routines on the inbound data definitions. The user would also create an Edibasic condition at the document level, in the original output document map, not to generate the document when the flag is set for reject. The user would then execute the acknowledgment map first, setting a status flag. If the inbound document is to be rejected, the outbound document will not be generated, based on the specified condition. Users can generate this type of acknowledgment at the document level only.

Purpose of Acknowledgments

The purpose of an acknowledgment is determined by the standard. The purpose is an indication of the way the acknowledgment should work. It is important to understand the various purposes of acknowledgments, so you can create your own based on your needs.

There are 3 basic types of acknowledgments:

- *Receipt acknowledgment*, to indicate whether an interchange has been received,
- *Compliance acknowledgment*, to indicate whether the syntax of the information that was sent complied with the syntactic requirements of the standard, which also implies receipt, or
- *Application acknowledgment*, a business information acknowledgment to indicate whether the content of the business information was correct.

Levels of Acknowledgment Response

You can specify acknowledgments to respond to any one of three levels of information:

- Interchange (Level 1)
- Functional group (Level 2)
- Document (Level 3)

Receipt acknowledgments are typically configured at the highest level, the interchange level (1).

Application acknowledgments must always be configured at the document level (3).

Acknowledgments Use Statuses

The method by which an acknowledgment conveys its message is by encoding the information in status values and associated information. To configure proprietary acknowledgments, you must understand statuses. Each standard that generates an acknowledgment has its own set of statuses. MW Translator also has an internal set of statuses to and from which it must convert the statuses of a given standard.

Internal Status Values

MW Translator uses internal status values that must be cross-referenced to those values used by a specific standard. When you generate outbound statuses for an acknowledgment to your trading partner, the TRM must know if it has to convert the internal statuses, which are numeric, to other values used by the outbound standard.

The status values may vary by standard. For example, The EDIFACT CONTRL that returns a status 8 simply implies that the interchange was received. On the other hand, the X12 TA1 returned status might be A (accepted), E (accepted with errors), or R (rejected). The purpose of both acknowledgments is to verify the receipt of an interchange, but the TA1 provides additional status information about the wrapper.

The internal status values are numeric. They are, in fact, a superset of those used for EDIFACT. The X12 standard, on the other hand, uses alpha values. The X12 values must be converted to and from the internal numeric values using cross-references.

The reconciliation process uses the numeric values listed in the following table. You should use these values for the output of a cross-reference table if the input values returned in the acknowledgment are different, as shown in the following Cross Reference window.

Internal Value	Description
1	Accepted
2	Accepted with errors
3	Partially accepted
4	Rejected
7	Acknowledged
8	Received
9	Interchange wrapper accepted
10	Interchange wrapper rejected
11	Interchange wrapper accepted with errors
12	Security reject

For X12, the following table is then used to convert the values. You can use the Edibasic functions *Xref\$* and *XrefR\$* (on page 759) and to do a regular or a reverse lookup in a table, respectively.

Input	Output
A	1
E	2
M	12
P	3
R	4
X	12

How Statuses Are Assigned

Statuses are assigned based on propagation and precedence rules. Typically, statuses are set by taking the highest-level status received and propagating that status to all lower levels, unless a more specific, detailed status is received for a lower level.

The rules are as follows:

Rule Type	Description
Propagation	<p>Statuses usually propagate down from the highest level, interchange, to the lowest level, document.</p> <p>Statuses can propagate up if a higher-level status exists of Awaiting ack or Ack not expected when a more detailed acknowledgment is received. In this case, the higher-level status will be changed to Detail ack received.</p>
Precedence	<p>The explicit status of a lower level (higher number) acknowledgment will replace the implicit status of a higher-level acknowledgment. A higher-level status will not replace a lower level status.</p> <p>An acknowledgment with a status of Received will be replaced by a later acknowledgment of the same level with any other status.</p> <p>In all other cases, where the acknowledgment level is identical, and the first acknowledgment does not have status of Received, the last acknowledgment received is retained.</p> <p>Manual acknowledgments override all other statuses and use the same propagation and precedence rules cited here.</p>

Special Cases for Trade Agreements

The following topics provide additional information about how to create trade agreement profiles for special requirements. Some instructions refer to the MW Translator Operator program. For more information about the MW Translator Operator program, refer to the *MW Translator Operator Guide and Reference*.

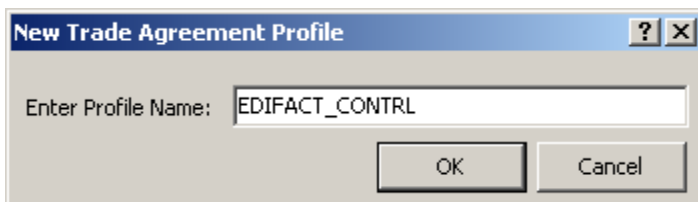
Configuring a Trade Agreement to Process an Acknowledgment without Mapping

The MW Translator Operator program has a selectable feature that will reconcile documents you send to your trading partner with a responding acknowledgment from your trading partner. If you want to use the reconciliation process, you need to be able to process inbound acknowledgments as if they were translated. However, you may not want to create a map and generate output. The following instructions show you how to do this.

To Configure a Trade Agreement to Process an Inbound Acknowledgment without Mapping

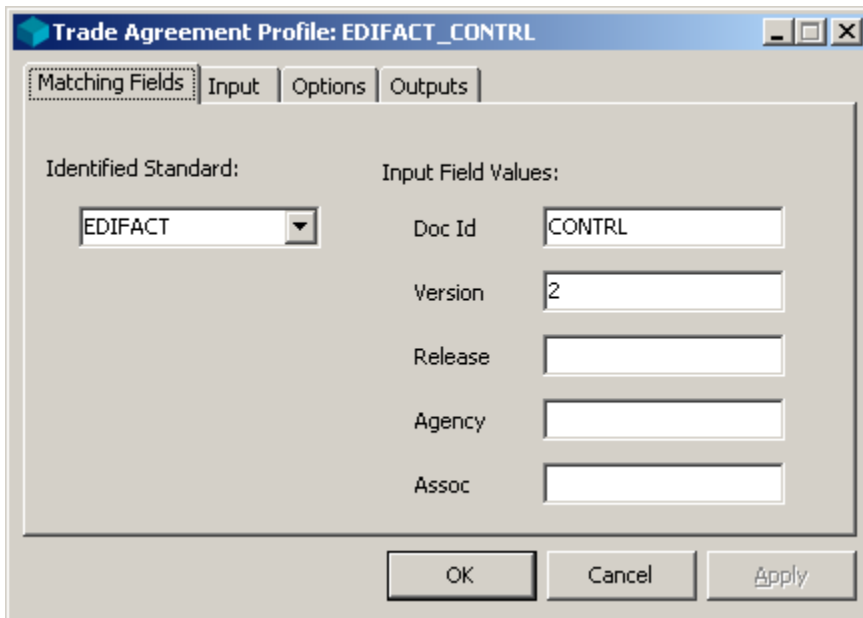
Use the Workbench to perform the following instructions. Make sure you are pointing to the database environment that you are using for the Operator program.


- 1 From Data Explorer select **Trade Agreements**, and then right-click and select **Add** from the menu. The **New Trade Agreement Profile** dialog box appears.
- 2 Enter a name for your trade agreement, and select **OK**.



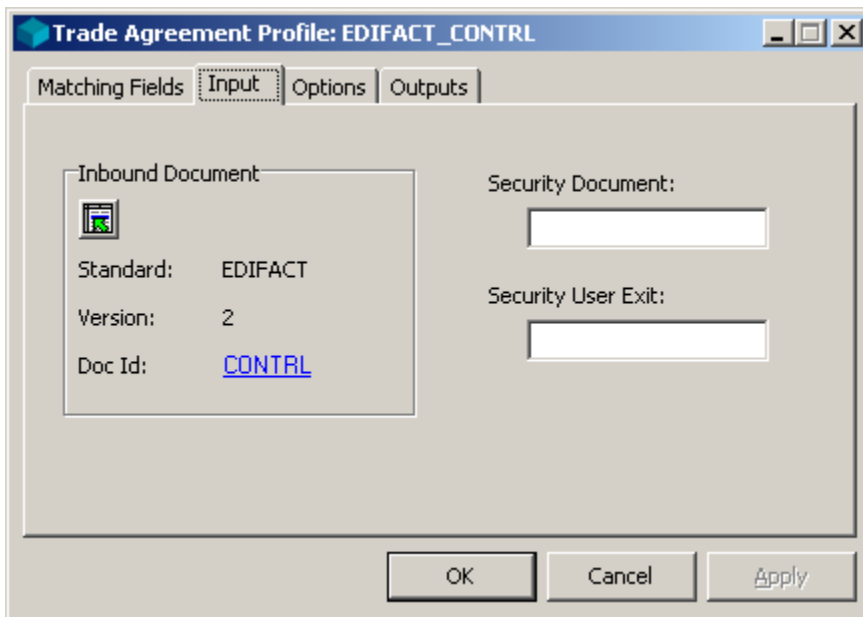
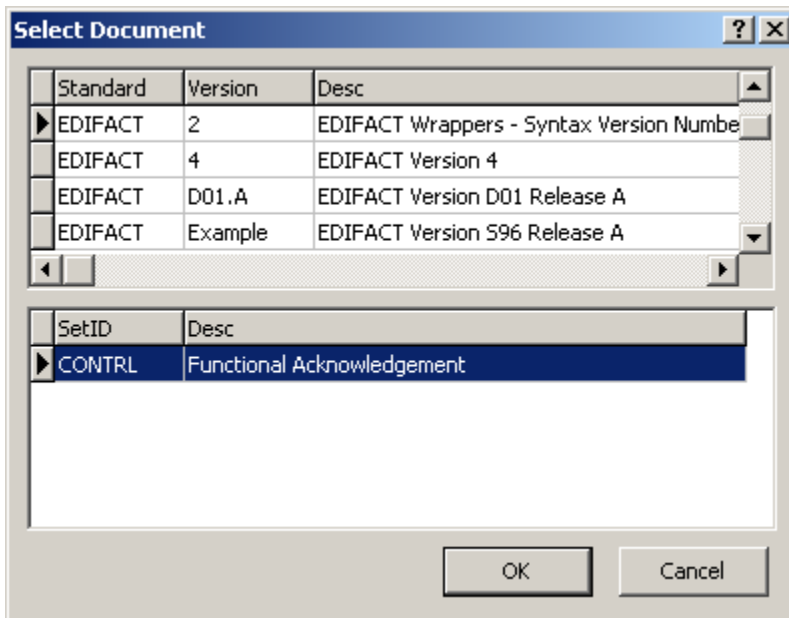
The Trade Agreement Profile window appears.

- 3 On the **Matching Fields** tab, enter the name of the standard, the acknowledgment, and the version.



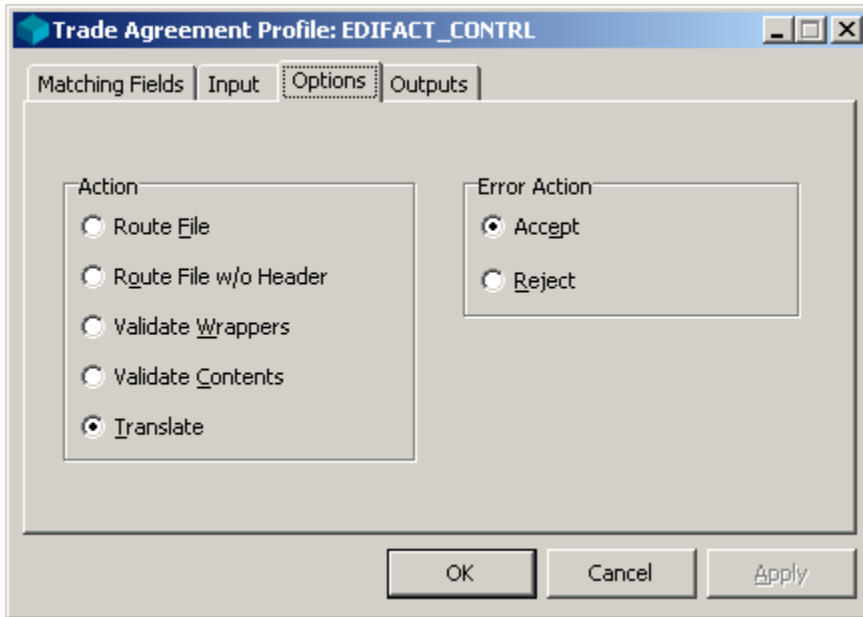
- 4 On the **Input** tab, click the **Select Source Doc** button .
The Select Document window appears.
- 5 Select the acknowledgment for the standard version you are using, and select **OK**.

The **Input** tab reappears with the configurations you selected.



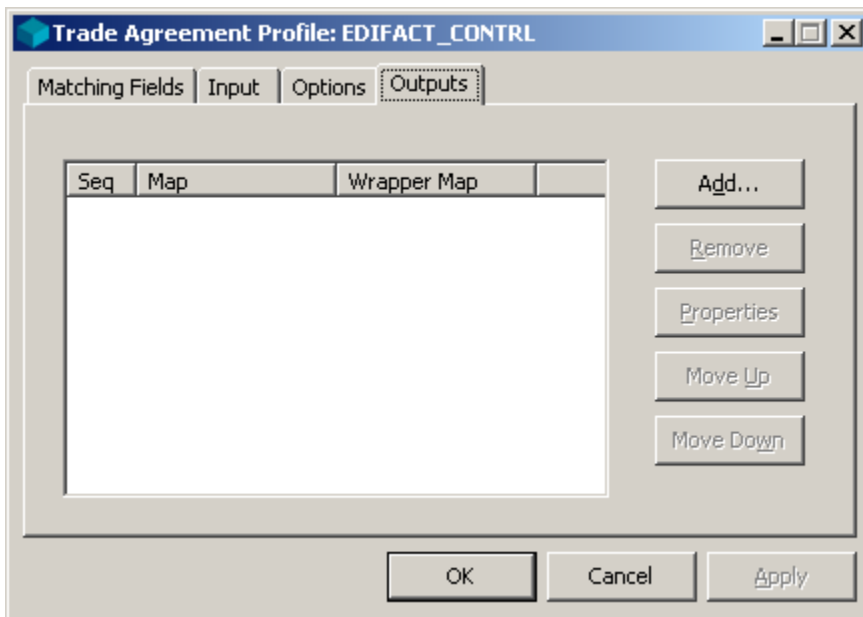
- 6 On the **Options** tab, select **Translate** and **Accept**.

You must select the translate action and the accept error action in order to have the TRM capture information for reconciliation.



- 7 Select **OK** to save your trade agreement profile.

You do not have to enter anything on the **Outputs** tab, unless you plan to create a map to save the information from the inbound acknowledgment.



Defining Partners

Overview of Defining Partners

There are various ways to control processing. Using partner definitions to control processing is one option. For a discussion about the options that are available and guidelines on which option to choose, refer to the section, *Planning and Configuration* (on page 11).

You control processing by determining to which definition you attach the trade agreement: Standard ID, Partner or Partner Relationship. When you associate the trade agreement with the Standard ID definition, any incoming data that matches a wrapper definition will use this trade agreement, which tells the TRM how to process the incoming data. When you associate the trade agreement with the recipient Partner definition, the incoming data must match the partner IDs of the recipient partner. When you associate the trade agreement with the Partner Relationship definition, the incoming data must match the partner IDs of the recipient partner and the sending partner. As you can see, the choices become more restrictive.

You may want to control processing using partner definitions for one of several reasons.

There are 3 primary reasons to define partners:

- **Security:** to control who is trading with whom by providing IDs for matching and optionally specifying that sender or recipient partner records must exist
- **Targeted output information:** to provide alternative IDs or routing locations for the outbound data
- **Targeted processing requirements:** to control processing by associating a trade agreement profile with a recipient partner or an acknowledgment profile with a sending partner, as individual partner definitions or part of a partner relationship definition

If you have specific processing requirements for the sender and recipient, you may need to define a relationship between the two partners.

If you have specific processing requirements for a group of partners to trade among themselves, you may need to define a closed group.

You can find more information about partners in the following sections:

- *Quick Configuration Tour* (on page 29)
- *Understanding Processing Flow* (on page 45)

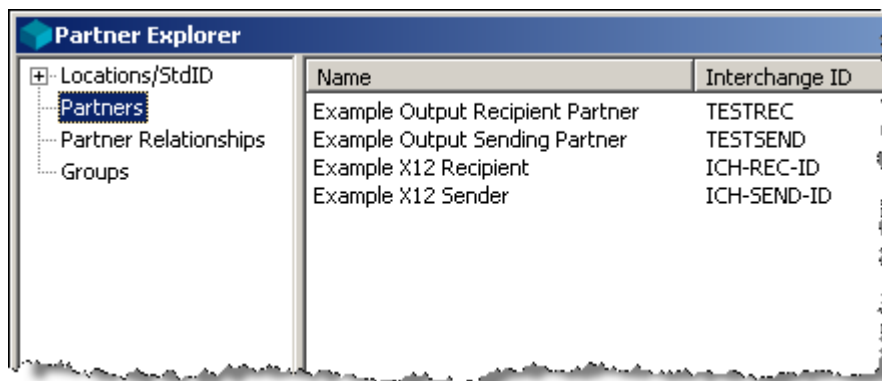
Using Partner Explorer or the Partner Wizard

You access those definitions associated with partners using Partner Explorer.

Partner Information

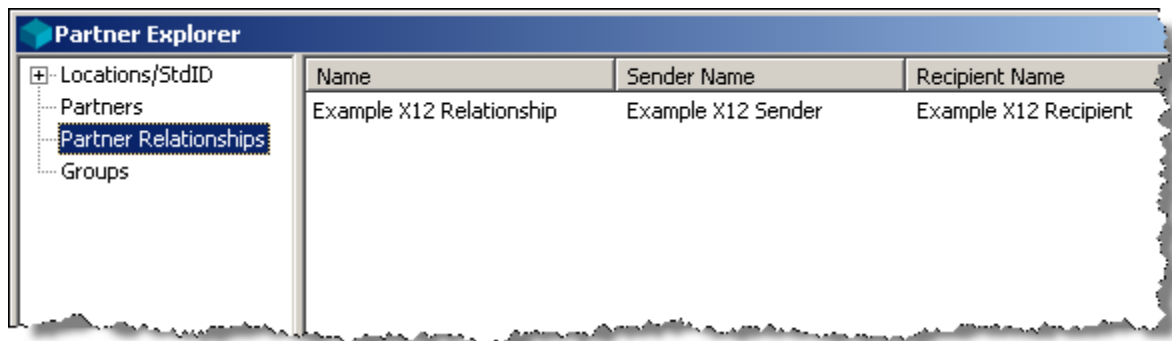
Partner definitions are organized by individual partners, partner relationships, and groups.

The **Partners** folder lists all partners you have defined, some of which may belong to a relationship or a group. Whenever you define a relationship or a group using partners that have not yet been defined, the new partner definitions also appear in the **Partners** folder.



Partner Explorer	
Name	Interchange ID
Example Output Recipient Partner	TESTREC
Example Output Sending Partner	TESTSEND
Example X12 Recipient	ICH-REC-ID
Example X12 Sender	ICH-SEND-ID

The **Partner Relationships** folder lists those specific partners that have a special relationship. This means that both the sender and recipient definitions are defined and visible in the **Partners** folder.

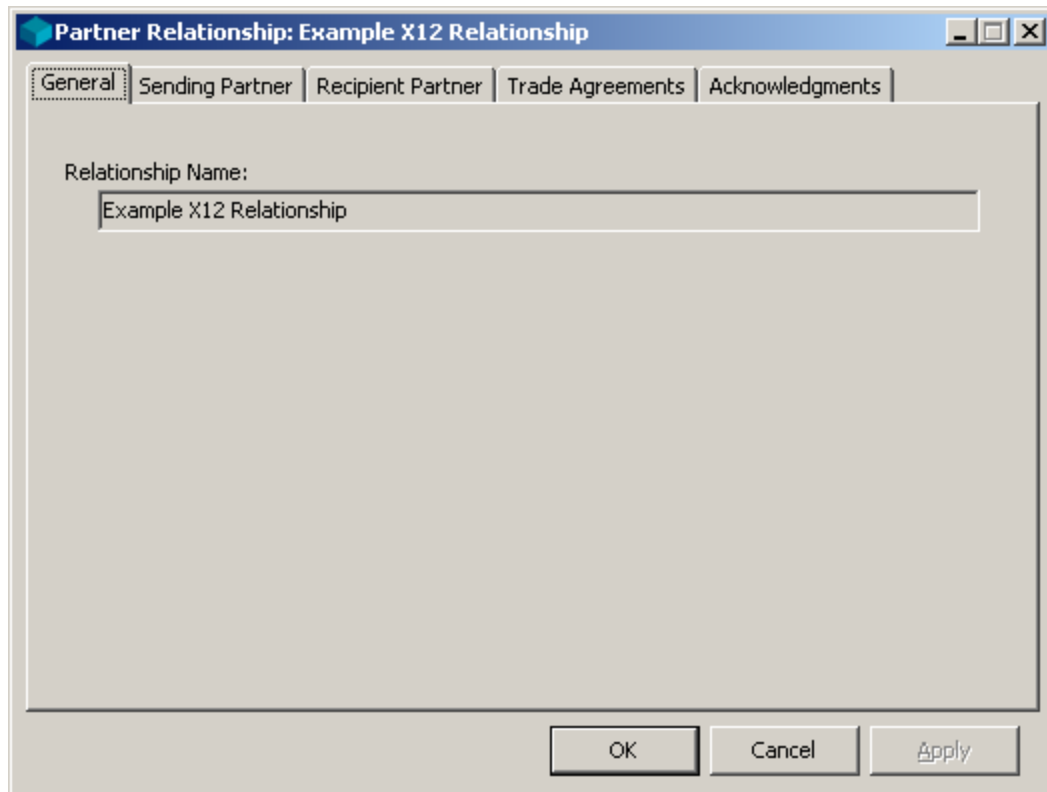


Partner Explorer		
Name	Sender Name	Recipient Name
Example X12 Relationship	Example X12 Sender	Example X12 Recipient

The **Groups** folder lists those partners who may trade among themselves.

Partner Relationships

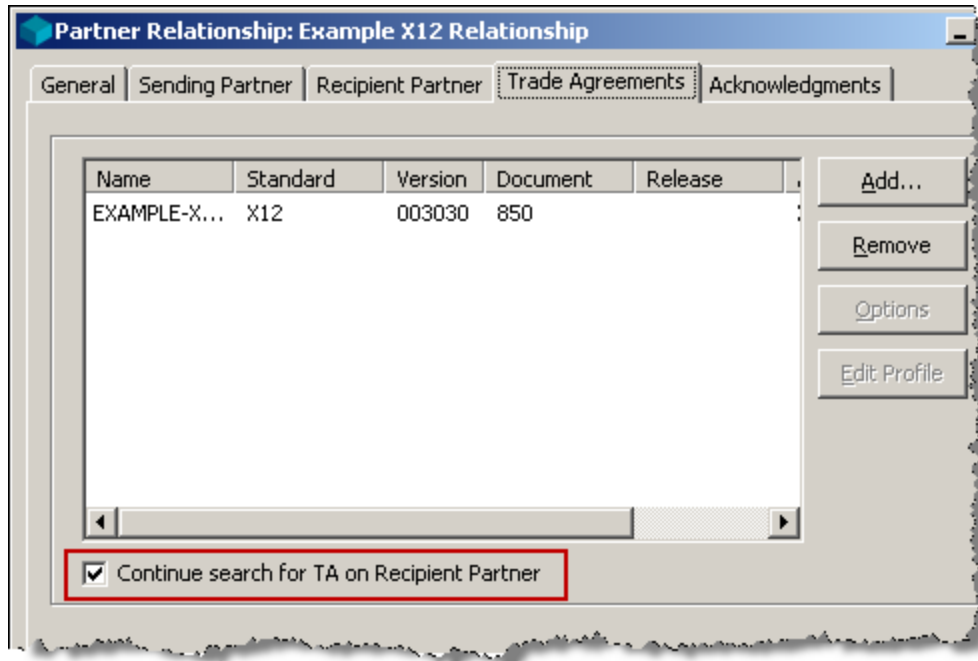
If two partners belong to a relationship, the search for configurations that control processing begins here. If there is any trade agreement or location information missing and **Continue Search for TA on Recipient Partner** is checked, the search continues with the recipient partner, then aliases, and finally, standard ID.



The screenshot shows a dialog box titled "Partner Relationship: Example X12 Relationship". The dialog has a tabbed interface with four tabs: "General", "Sending Partner", "Recipient Partner", "Trade Agreements", and "Acknowledgments". The "General" tab is currently selected. Inside the dialog, there is a label "Relationship Name:" followed by a text input field containing the text "Example X12 Relationship". At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Apply".

Alternate IDs and Locations for Outputs

You can directly specify alternative IDs and locations when you define your partners or partner relationships. You can add, review or change your alternative information by selecting the **Options** button for the appropriate trade agreement from the **Trade Agreements** tab on either the Partner or Partner Relationship window.



You specify alternative information for the sending partner of the outbound document on the **Sending Partner** tab: alternative IDs and an alternative source location, which you may override with the Override Location.

The screenshot shows a dialog box titled "Partner Relationship Trade Agreement Options: EXAMPLE-X850". At the top, there is an "Output:" dropdown menu with the value "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner" (selected), "Recipient Partner", and "Misc".

Under the "Sending Partner" tab, there are several input fields:

- Partner Name:** A dropdown menu with the value "Example Output Sending Partner" highlighted by a red box.
- Location:** An empty text input field.
- Override Location:** A text input field containing the value "TESTSEND-MAILBOX".

There are two sections for defining identifiers:

- Interchange:** Contains an "ID:" field with the value "TESTSEND" (highlighted by a red box), a "Qual:" field, and "Int ID:" and "Int ID2:" fields.
- Functional Group:** Contains an "ID:" field, a "Qual:" field, and "Int ID:" and "Int ID2:" fields.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

You specify alternative information for the recipient partner of the outbound document on the **Recipient Partner** tab: alternative IDs and an alternative destination location, which you may override with the Override Location.

The screenshot shows a dialog box titled "Partner Relationship Trade Agreement Options: EXAMPLE-X850". At the top, there is an "Output:" dropdown menu with the value "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner", "Recipient Partner" (which is selected), and "Misc".

Under the "Recipient Partner" tab, there are several input fields:

- "Partner Name:" dropdown menu with the value "Example Output Recipient Partner" (highlighted with a red box).
- "Location:" empty text field.
- "Override Location:" text field with the value "TESTREC-MAILBOX".

There are two sections for IDs:

- Interchange:** "ID:" text field with the value "TESTREC" (highlighted with a red box), "Qual:" empty text field, "Int ID:" empty text field, and "Int ID2:" empty text field.
- Functional Group:** "ID:" empty text field, "Qual:" empty text field, "Int ID:" empty text field, and "Int ID2:" empty text field.

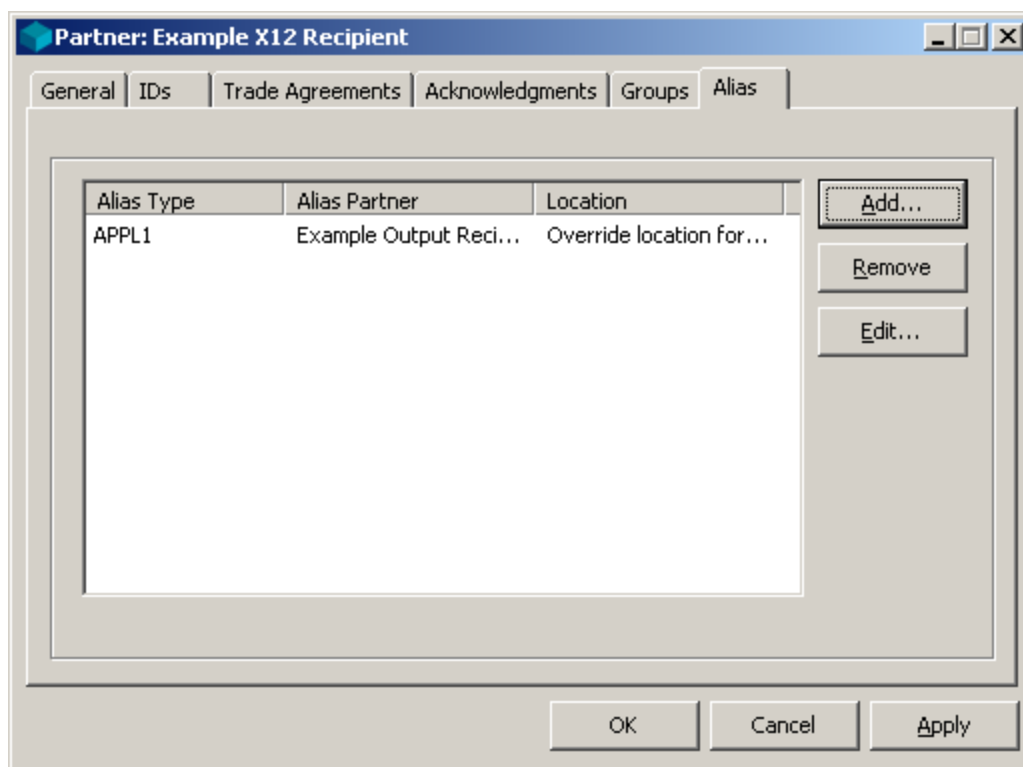
At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Aliases

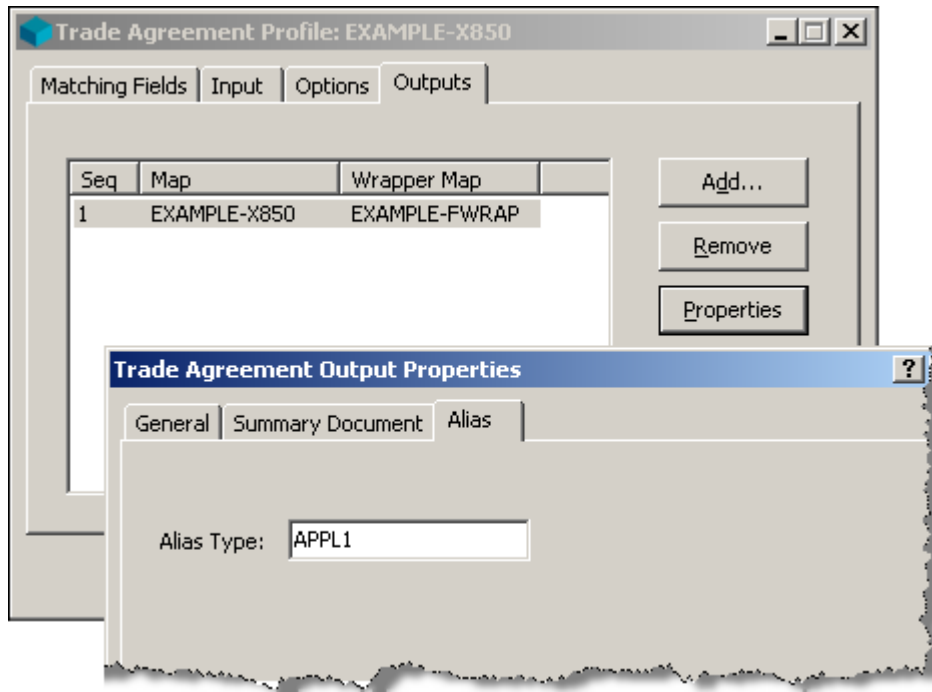
Trade agreement properties, however, do not provide the flexibility of aliases, since the aliases can be used when you are missing one of the partner definitions.

The search for aliases begins when required information is not found on Partner Relationship or Partner definitions or on the Trade Agreement Properties for the partner relationship or the partner. Aliases are checked before the default routing on standard ID.

NOTE: Aliases are not bidirectional definitions. You must set up the links separately for each partner alias definition.



The trade agreement outputs must also specify the alias, so that when it is invoked, it will trigger the use of the aliased partner information. You access Trade Agreement Profile windows from Data Explorer.



Using the Partner Wizard

The Partner Wizard will lead you through the maze of setting up new partnerships and locations and standard IDs. If you need to modify definitions, you must do so directly from Partner Explorer.

Once you have been led through the definitions process, the wizard displays a list of definitions that it will use for the new partnership for you to review.

For a discussion of using the Partner Wizard to create partner-related definitions, refer to the topic *Creating Definitions Using the Partner Wizard* (on page 290).

Configuring Partner Information

Identifying partners is part of partner-based processing control. Though you may not want to specify a trade agreement for a recipient partner, you may still want to define the partner as a security measure. Remember that the trade agreement must be found somewhere during processing. If it is not associated with the recipient partner, then it must be associated with the incoming wrapper in the inbound location for Standard ID.

Whether or not a partner relationship or partner definition is associated with a trade agreement profile or an acknowledgment profile, it can still provide the following functionality:

- Ensure that partners who send and receive data are known entities
- Specify IDs for outbound wrappers
- Specify routing locations

To configure partner matching, you have two steps:

- 1 Associate elements in wrapper definitions with internal fields
- 2 Define the partner relationship or partner definitions using the appropriate IDs

Using Internal Fields for Partner Information

You can use eight internal fields for partner information. You can use them at any of the possible levels of wrapper definitions. They can be used to match partner definitions. The fields and their typical uses are as follows.

Internal Field	Use	Example Where Used
Sender Partner Id	Match sending partner	X12 ISA.06 EDIFACT UNB.2:1
Sender Partner Qual	Match sending partner qualifier	X12 ISA.05 EDIFACT UNB.2:2
Sender Partner Int ID	Sending partner internal ID	EDIFACT UNB.2:3
Sender Int ID2	Sending partner internal ID2	
Rec Partner Id	Match recipient partner	X12 ISA.08 EDIFACT UNB.3:1
Rec Partner Qual	Match recipient partner qualifier	X12 ISA.07 EDIFACT UNB.3:2
Rec Partner Int ID	Recipient partner internal ID	EDIFACT UNB.3:3
Rec Partner Int ID2	Recipient partner internal ID2	

Associate Elements with Internal Fields

The purpose of associating wrapper element definitions with internal fields for partners is so that when the inbound wrapper is parsed, the information is stored in the appropriate field. The search for a matching partner depends on the values in the inbound wrapper having been properly stored in the internal fields.

IMPORTANT: If partner fields are not defined on the wrapper or if the corresponding element is omitted in the inbound data, then the internal field will be initialized to a null string.

This step has already been done for public standard wrappers. You must do this for any other wrapper definitions you create. Consider the following examples.

The EDIFACT wrapper without functional groups has partner IDs for the interchange level only. This window shows you the internal fields for partner information associated with the elements of the EDIFACT composite elements S002 and S003 in the interchange level segment, the UNB.

Composite: EDIFACT, 2, S002

Description: INTERCHANGE SENDER

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	0004	Sender identification	Send Partner ID	M	AN	1	35
2	0007	Partner identification code qualifier	Send Partner Qual	C	AN	1	4
3	0008	Address for reverse routing	Send Partner IntID	C	AN	1	14

Composite: EDIFACT, 2, S003

Description: INTERCHANGE RECIPIENT

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	0010	Recipient identification	Rec Partner ID	M	AN	1	35
2	0007	Partner identification code qualifier	Rec Partner Qual	C	AN	1	4
3	0014	Routing address	Rec Partner IntID	C	AN	1	14

Since the X12 envelope has both interchange and functional group level wrappers, and since there are partner IDs for both levels, you will see the internal fields for partner information assigned at both levels. This window shows you the internal fields for partner information associated with the elements of the X12 interchange level segment, the ISA.

Segment: X12, 005010, ISA

Description: Interchange Control Header

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	I01	Authorization Information Qualifier		M	ID	2	2
2	I02	Authorization Information		M	AN	10	10
3	I03	Security Information Qualifier	Password Qual	M	ID	2	2
4	I04	Security Information	Password	M	AN	10	10
5	I05	Interchange ID Qualifier	Send Partner Qual	M	ID	2	2
6	I06	Interchange Sender ID	Send Partner ID	M	AN	15	15
7	I05	Interchange ID Qualifier	Rec Partner Qual	M	ID	2	2
8	I07	Interchange Receiver ID	Rec Partner ID	M	AN	15	15
9	I08	Interchange Date	Date	M	DT	6	6
10	I09	Interchange Time	Time	M	TM	4	4
11	I65	Repetition Separator	Repetition Separator	M	DL	1	1
12	I11	Interchange Control Version Number	Interchange Version	M	ID	5	5
13	I12	Interchange Control Number	Control Reference	M	NO	9	9
14	I13	Acknowledgment Requested	Acknowledgement Flag	M	ID	1	1
15	I14	Interchange Usage Indicator	Test Indicator	M	ID	1	1

This window shows you the internal fields for partner information associated with the elements of the X12 functional group level segment, the GS.

Segment: X12, 005010, GS
Description: Functional Group Header

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	479	Functional Identifier Code	Functional Group ID	M	ID	2	2
2	142	Application Sender's Code	Send Partner ID	M	AN	2	15
3	124	Application Receiver's Code	Rec Partner ID	M	AN	2	15
4	373	Date	Date	M	DC	8	8
5	337	Time	Time	M	TM	4	8
6	28	Group Control Number	Control Reference	M	NO	1	9
7	455	Responsible Agency Code	Agency	M	ID	1	2
8	480	Version / Release / Industry Identifier Code	Document Version	M	AN	1	12

This window shows you the internal fields for partner information associated with the elements of a proprietary wrapper segment. Notice that there is only one element here associated with a partner field. This is for the recipient partner. Since there is no sending partner identified, presumably the sending partner is always the same partner, such as ABC Company. Your requirements may vary.

Segment: EXAMPLE, 1, ***
Description: Fixed Header Record

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	0008	Record Tag (3 character)		M	AN	1	3
2	0002	Partner	Rec Partner ID	M	AN	1	12
3	0201	Transaction Set ID	Document ID	M	AN	1	3

Defining Partner IDs for Partners

When you create Partner Relationships, Closed Groups, or Partners, or you specify partners on the Trade Agreement Options window, partner definitions are created in the **Partners** folder. The Partner Relationship definition adds another layer of definitions for a particular trading partnership between two partners, which is stored in the **Partner Relationships** folder.

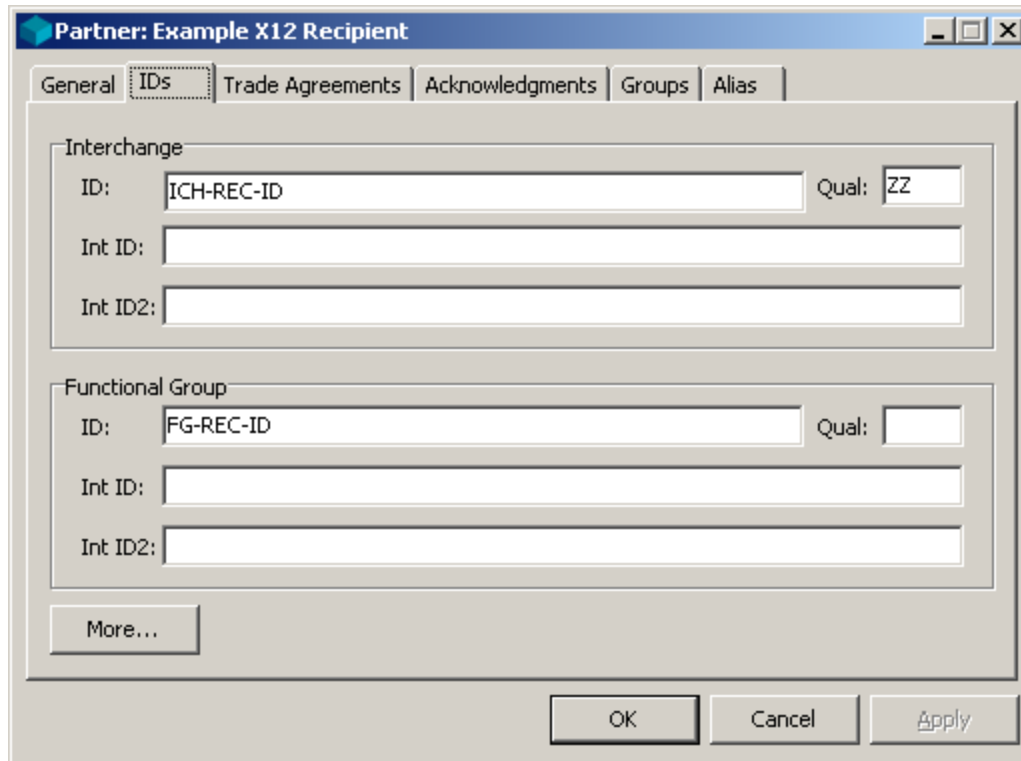
Partner definitions in the **Partners** folder can be used as either the sender or the recipient for alternative IDs or locations. How it is used might be determined at runtime, as it is for partners, or it might be determined by the definition, as it is for a partner relationship.

However they are used, partner definitions are the basis for partner-based processing. Therefore, let us look at some sample definitions of partners.

This partner definition uses only one ID. To be selected, this partner ID must match the value parsed from the incoming wrapper. The element that defines this value, must also be associated with the **Send Partner ID** or **Rec Partner ID** internal fields.

The screenshot shows a dialog box titled "Partner: Example Output Recipient Partner". It has several tabs: "General", "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "IDs" tab is selected. The dialog is divided into two main sections: "Interchange" and "Functional Group". Each section contains three input fields: "ID:", "Int ID:", and "Int ID2:". The "ID:" field in the "Interchange" section is filled with the text "TESTREC". To the right of each "ID:" field is a "Qual:" field. At the bottom of the dialog, there are four buttons: "More...", "OK", "Cancel", and "Apply".

This partner definition uses two partner values at the interchange level, and one at the functional group level. In fact, this partner definition matches the recipient partner in the X12 wrapper of the example delivered with the Workbench.



Partner: Example X12 Recipient

General | **IDs** | Trade Agreements | Acknowledgments | Groups | Alias

Interchange

ID: ICH-REC-ID Qual: ZZ

Int ID:

Int ID2:

Functional Group

ID: FG-REC-ID Qual:

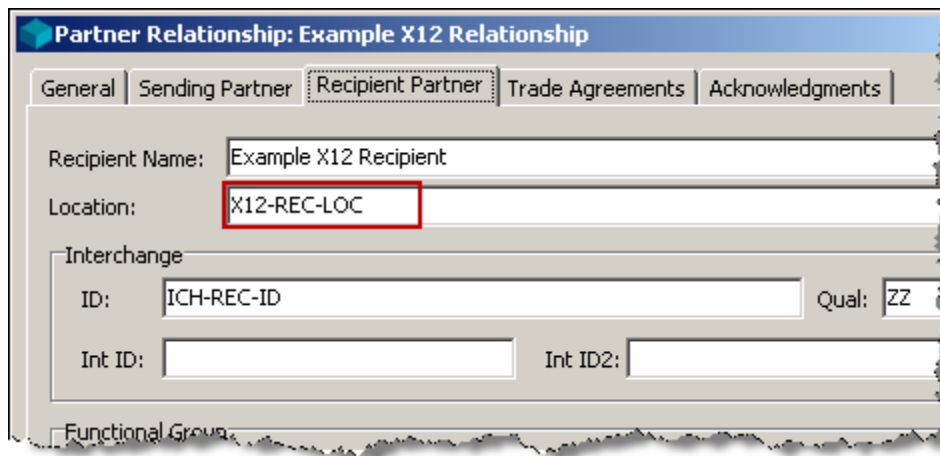
Int ID:

Int ID2:

More...

OK Cancel Apply

This partner is also used in a partner relationship, as shown in the following window.



Partner Relationship: Example X12 Relationship

General | Sending Partner | **Recipient Partner** | Trade Agreements | Acknowledgments

Recipient Name: Example X12 Recipient

Location: X12-REC-LOC

Interchange

ID: ICH-REC-ID Qual: ZZ

Int ID: Int ID2:

Functional Group

Configuring Partners to Control Processing

In order to configure processing control based on partner definitions, you want to ensure the following:

- The incoming partner IDs match existing definitions, and
- The correct trade agreement profile is selected, and
- If generating acknowledgments, the correct acknowledgment profile is selected

For comparison, in order to configure processing control based on the incoming wrapper of Standard ID, you want to ensure the following:

- NO incoming partner ID matches existing definitions, or
- A matching trade agreement is not on the list for partner relationship or partner definitions, and
- A matching trade agreement is on the list for incoming wrapper of standard ID

Specifying Alternative Output IDs or Routing

Trading partners may use any one of several IDs to identify themselves. For each unique combination of IDs by which they are known, they can have a partner record. By linking the IDs, you can specify that an incoming ID or location be changed to a different ID or location for the output. This would be the case if you do not want to use the IDs or locations of the incoming partners. There are two ways to do this using partner definitions:

- Directly during generation as part of the partner definition using the Trade Agreement Options window
- Indirectly during generation using aliases to link related partner definitions

NOTE: If you do not provide alternative IDs or locations using one of these methods, the IDs and locations for the incoming partners are used by default.

Using Trade Agreement Options

You can use the Trade Agreement Options window to specify alternative IDs for the output wrapper or source or destination locations for the output. The Trade Agreement Options window is directly tied to the recipient partner. To be able to correctly use alternative IDs or locations this way, the sending partner must always be associated with the recipient partner, or the recipient partner must be the only ID you want to change. Therefore, you would use this method to link partners for the following configurations:

- Partner relationship (sender is always linked to recipient for this trade agreement)
- Recipient partner only (sender ID will not be changed in output, if used)

In fact this method is used to provide alternative IDs and locations for the output for the partner relationship defined in the EXAMPLE translation.

Consider the following definition for the sending partner. Initially, the IDs and location are the same as in the partner record. You can override only the location. This location override value will not be reflected in the partner record.

The screenshot shows a dialog box titled "Partner Relationship Trade Agreement Options: EXAMPLE-X850". At the top, there is an "Output:" dropdown menu with the value "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner" (selected), "Recipient Partner", and "Misc".

Under the "Sending Partner" tab, there are several fields:

- Partner Name:** A dropdown menu with the value "Example Output Sending Partner".
- Location:** An empty text input field.
- Override Location:** A text input field containing the value "TESTSEND-MAILBOX".

Below these fields is a section titled "Interchange" with the following fields:

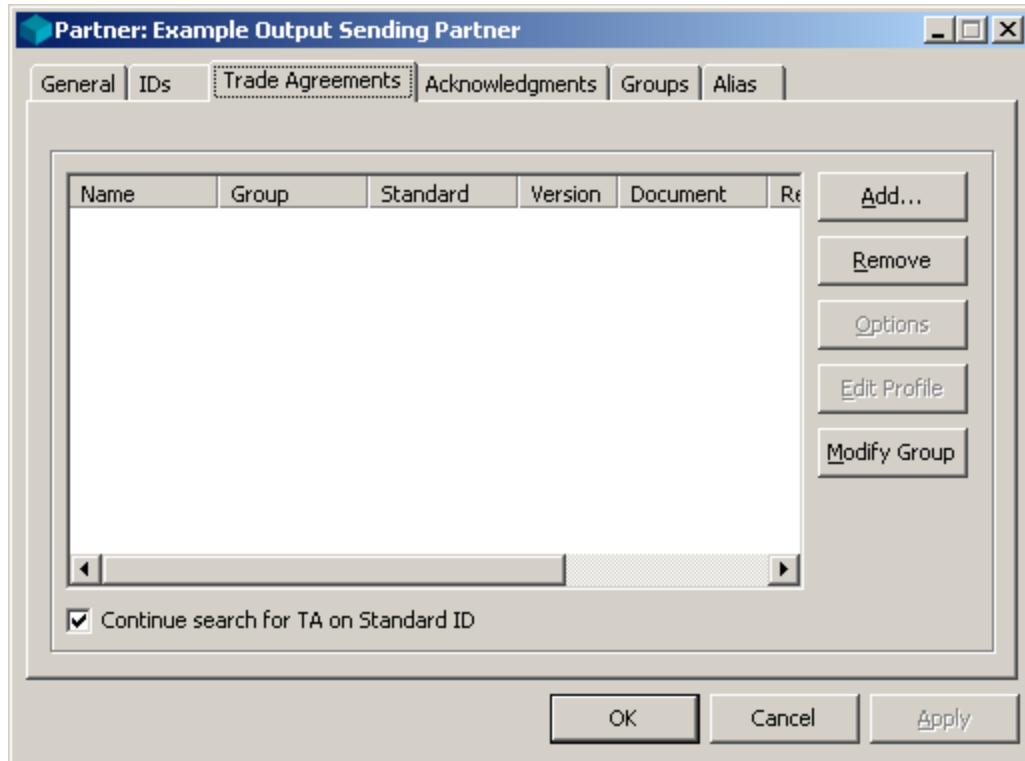
- ID:** A text input field containing the value "TESTSEND".
- Qual:** An empty text input field.
- Int ID:** An empty text input field.
- Int ID2:** An empty text input field.

At the bottom of the dialog is a section titled "Functional Group" with the following fields:

- ID:** An empty text input field.
- Qual:** An empty text input field.
- Int ID:** An empty text input field.
- Int ID2:** An empty text input field.

At the very bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Note that these values are the same as seen in the Partner record for this partner. In fact, if the partner does not exist, it will be created in the **Partners** folder.



The screenshot shows a dialog box titled "Partner: Example Output Sending Partner". It has a tabbed interface with the following tabs: "General", "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "IDs" tab is currently selected. The dialog is divided into two main sections: "Interchange" and "Functional Group".

Interchange Section:

- ID: Qual:
- Int ID:
- Int ID2:

Functional Group Section:

- ID: Qual:
- Int ID:
- Int ID2:

At the bottom left of the dialog is a "More..." button. At the bottom right are three buttons: "OK", "Cancel", and "Apply".

Consider the following definition for the recipient partner. Initially, the IDs and location are the same as in the partner record. You can override only the location. This location override value will not be reflected in the partner record.

Partner Relationship Trade Agreement Options: EXAMPLE-X850 ? X

Output: 1,EXAMPLE,1,FPO 1 of 1

Sending Partner Recipient Partner Misc

Partner Name: Example Output Recipient Partner

Location:

Override Location: TESTREC-MAILBOX

Interchange

ID: TESTREC Qual:

Int ID: Int ID2:

Functional Group

ID: Qual:

Int ID: Int ID2:

OK Cancel Apply

Note that these values are the same as seen in the Partner record for this partner.

The screenshot shows a dialog box titled "Partner: Example Output Recipient Partner". The "General" tab is selected. The "Partner Name" field contains the text "Example Output Recipient Partner". The "Location" field is empty. At the bottom, there are three buttons: "OK", "Cancel", and "Apply".

The screenshot shows the same dialog box, but with the "IDs" tab selected. It displays two sections: "Interchange" and "Functional Group". Each section has three input fields: "ID:", "Int ID:", and "Int ID2:". The "ID:" field in the "Interchange" section contains the text "TESTREC". The "Qual:" field next to it is empty. The "More..." button is located at the bottom left of the dialog. At the bottom, there are three buttons: "OK", "Cancel", and "Apply".

Using Aliases

You can also provide alternative IDs for the output wrapper and alternative locations by configuring aliases. Aliases allow you to specify alternative IDs for a sending partner, because the alias key is associated with the Trade Agreement Output itself, not the Trade Agreement Options of the recipient partner.

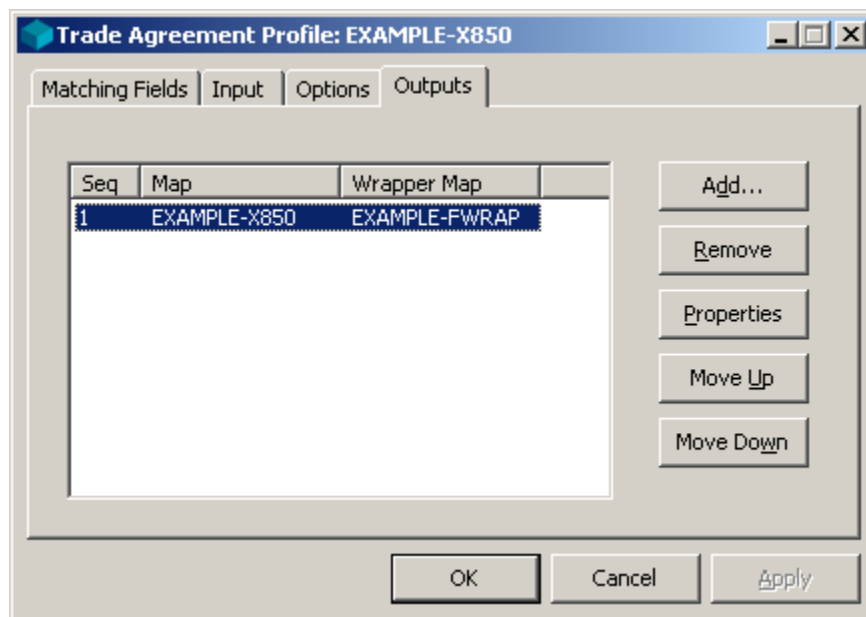
You would use aliases, rather than Trade Agreement Output Options, when the sending partner might be any one of your partners. The inbound partner must be defined before you specify the other alias configurations.

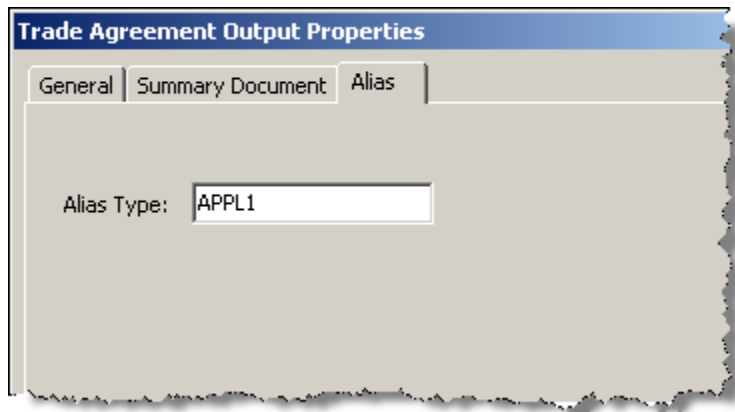
In order to use aliases, you configure the following:

- Trade Agreement Output Properties with an alias type
- An aliased partner for the sending partner, if desired
- An aliased partner for the recipient partner, if desired

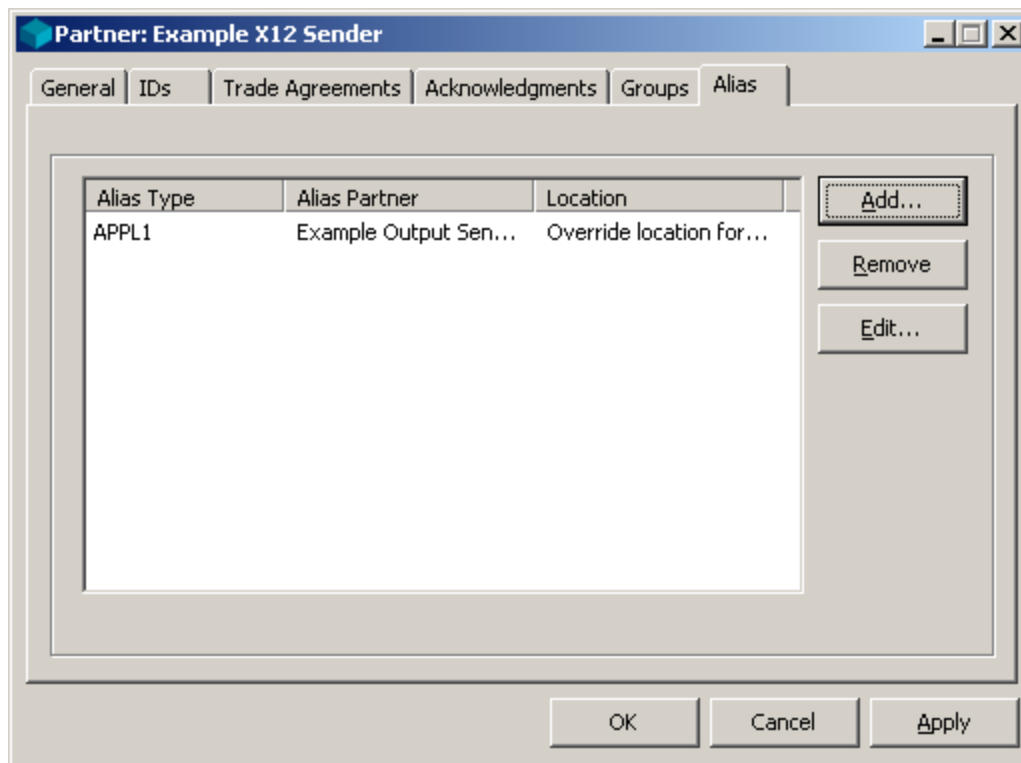
For example, assume that you have a partner record defined for an incoming sending partner. You want to convert that sending partner ID, which is used for the incoming X12 document, to an ID in the wrapper for the outbound document that your application requires. You will also use the location for the aliased partner as the source location of the output.

You must add the alias type to the trade agreement output properties configuration for the trade agreement that will produce the output, EXAMPLE-X850, as follows:





Then you link the inbound sending ID, Example X12 Sender, with the outbound sending ID, Example Output Sending Partner, by adding the outbound sending partner record associated with the alias type, APPL1, to the sending partner's alias list. For specific instructions, refer to the topic, *To Add an Alias to a Partner* (on page 785). This provides the link between the two partners, using a third piece of information, the alias ID.



When the output is generated, the ID from the aliased partner is used instead of the ID from the inbound partner.

The screenshot shows a dialog box titled "Partner: Example Output Sending Partner". It has several tabs: "General", "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "IDs" tab is selected. The dialog is divided into two main sections: "Interchange" and "Functional Group". Each section contains three input fields: "ID", "Int ID", and "Int ID2". To the right of the "ID" field in each section is a "Qual" field. In the "Interchange" section, the "ID" field contains the text "TESTSEND". At the bottom of the dialog, there is a "More..." button and three buttons: "OK", "Cancel", and "Apply".

By default, the location of the sending partner becomes the source location of the output. In this case, the source location is the override location specified for the alias.

Considering Additional Processing Controls

You can specify additional controls from Partner Explorer, which will give you a better idea of all your options.

Restricting Processing to Defined Partner

You can require that partner definitions be used for the sending and/or recipient partners by checking the appropriate box in the **Partner Definition Required** area on the **Partners** tab of the Standard ID window. Relatively early in the processing cycle, after the wrappers have been parsed, an attempt is made to find matching partner definitions. If the partner IDs parsed from the input data do not match any defined partners, processing terminates.

The screenshot shows a window titled "Standard ID: X850TEST, 1, X12, EXAMPLE, ISA". The "Partners" tab is selected. The "Partner Definition Required" section has two checked checkboxes: "Sending Partner" and "Recipient Partner". Below this are two sections for "Default Sending Partner" and "Default Recipient Partner", each with a "Name:" label and a dropdown menu. At the bottom are "OK", "Cancel", and "Apply" buttons.

Using Default Partner Definitions

If you are not required to match incoming definitions, and no matching definitions are found, it is possible to use default partner definitions.

Using this feature requires that either no partner information was in the wrapper, or that the partner information in the wrapper was not associated with the appropriate internal fields. If both the ID and Qual internal fields for the sending or recipient partner are null (string with a length of zero) in the internal table, then the corresponding default partner definition will be used.

For more information refer to the topic, *Step 4. Look for Defined Partner IDs for the Sender and Recipient at this Level* (on page 53).

Task 5. Creating Partner Definitions

This is where you can put all of your definitions to use. You have many ways to define your trade relationships. This section discusses how to configure both a partner relationship and individual partners using Partner Explorer. It also discusses how to use the Partner Wizard to define partner relationships or partners.

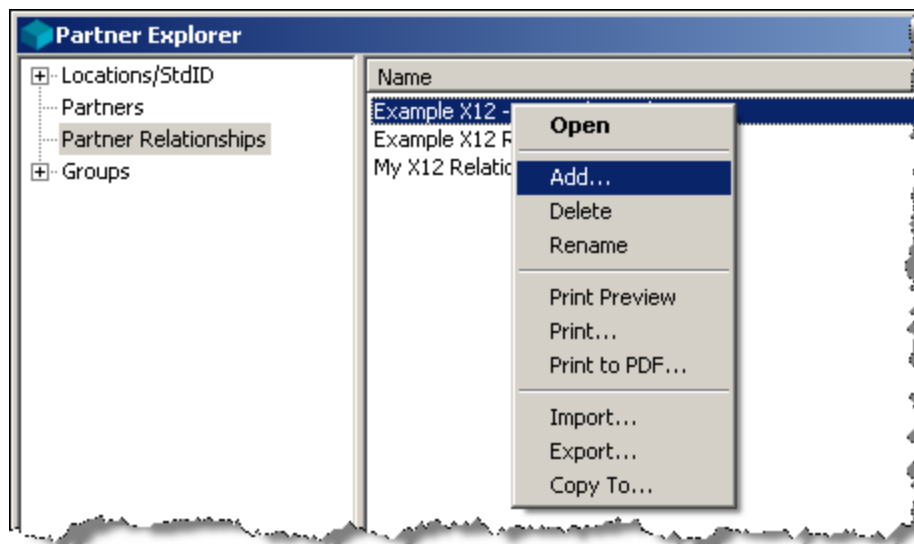
Task 5.1 creates a partner relationship between two individual partners. If the individual partners do not already exist, it creates them. Task 5.2 simply creates the individual partners. You would choose one or the other.

Task 5.1. Creating a Partner Relationship

A partner relationship defines a more restrictive trade agreement between two partners than the more generic individual partner definitions. Individual partner definitions can be used as either a sender or recipient, which is decided at runtime. The partner relationship definition specifies which is the sender and which is the recipient.

Task 5.1.1. Specifying Partner Relationship Name

You can create a partner relationship by selecting the **Partner Relationships** folder then **Add** from the **File** menu or right-click and **Add** from the menu. This creates a definition in the **Partner Relationships** folder.



You must also specify sending and recipient partner IDs to create the partnership, which you can do by selecting existing partner definitions or entering new partner names. If you enter new partner names, new partners are created in the **Partners** folder. A partner relationship, after all, is a strictly defined relationship between two existing partners, who may be trading in other ways as well as participating in this particular relationship.

Task 5.1.2. Specifying Partner IDs

The partners may exist, but you now must specify IDs for the sending and recipient partners to match with the incoming data. When both sets of IDs match, the relationship definition is used. You can also enter locations to be associated with this partner.

The screenshot shows a dialog box titled "Partner Relationship: Example X12 Relationship" with a blue header bar. It has four tabs: "General", "Sending Partner", "Recipient Partner", "Trade Agreements", and "Acknowledgments". The "Sending Partner" tab is selected. The dialog contains the following fields:

- Sender Name:** A dropdown menu with "Example X12 Sender" selected.
- Location:** A text box containing "X12-SEND-LOC".
- Interchange:** A section with two rows of fields:
 - Row 1: "ID:" followed by a text box containing "ICH-SEND-ID" and "Qual:" followed by a text box containing "ZZ".
 - Row 2: "Int ID:" followed by an empty text box and "Int ID2:" followed by an empty text box.
- Functional Group:** A section with two rows of fields:
 - Row 1: "ID:" followed by a text box containing "FG-SEND-ID" and "Qual:" followed by an empty text box.
 - Row 2: "Int ID:" followed by an empty text box and "Int ID2:" followed by an empty text box.

At the bottom left, there is a "More..." button. At the bottom right, there are three buttons: "OK", "Cancel", and "Apply".

Partner Relationship: Example X12 Relationship

General | Sending Partner | **Recipient Partner** | Trade Agreements | Acknowledgments

Recipient Name: Example X12 Recipient

Location: X12-REC-LOC

Interchange

ID: ICH-REC-ID Qual: ZZ

Int ID: Int ID2:

Functional Group

ID: FG-REC-ID Qual:

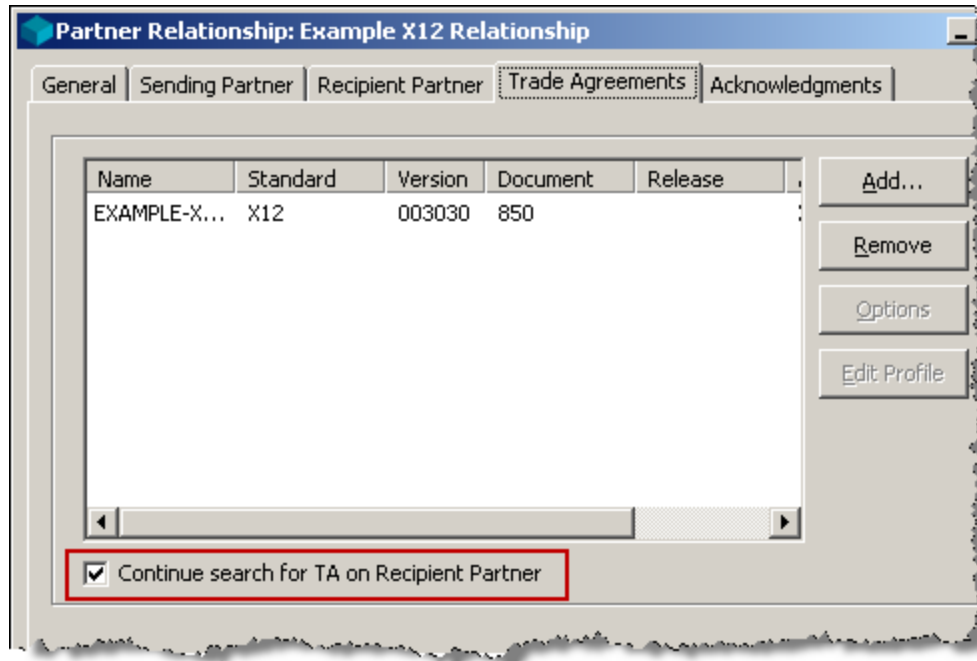
Int ID: Int ID2:

More...

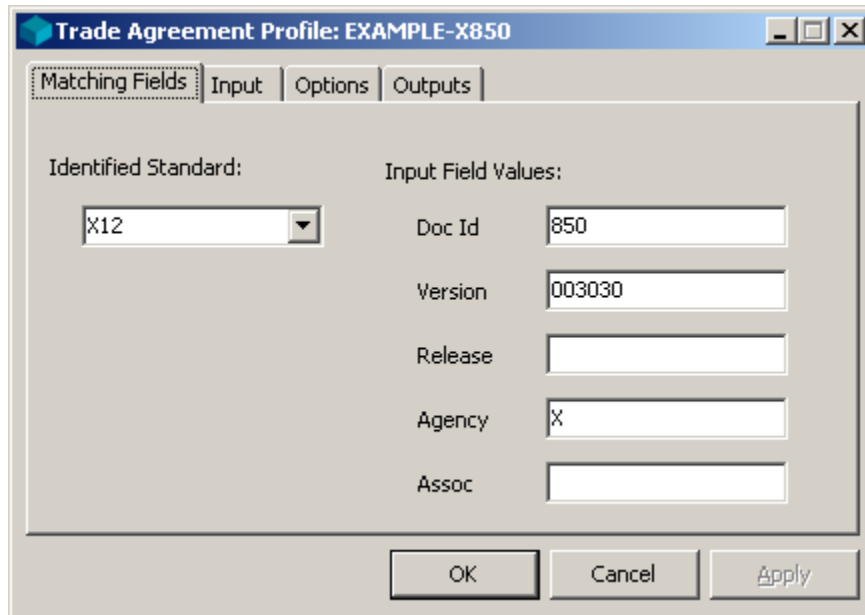
OK Cancel Apply

Task 5.1.3. Specifying the Trade Agreement Profile

Although you can do so, there is little reason to create a partner relationship without associating it with a trade agreement profile. One of the reasons you use partner relationships is to narrow the choice of the trade agreement profiles that might be used. You do this by adding trade agreement profiles to a list on the **Trade Agreements** tab.



An attempt is made to match each of the profiles with the inbound data, using the first profile that matches the **Input Field Values** on the Trade Agreement Profile itself. The standard identified during the initial phases of processing must match the value in the **Identified Standard** field.



The screenshot shows a dialog box titled "Trade Agreement Profile: EXAMPLE-X850". It has four tabs: "Matching Fields", "Input", "Options", and "Outputs". The "Matching Fields" tab is selected. The dialog is divided into two main sections: "Identified Standard:" and "Input Field Values:". Under "Identified Standard:", there is a dropdown menu showing "X12". Under "Input Field Values:", there are five text input fields: "Doc Id" with "850", "Version" with "003030", "Release" (empty), "Agency" with "X", and "Assoc" (empty). At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

If there is no trade agreement on the list that matches the inbound data, and if you have checked **Continue search for TA on Recipient Partner**, processing continues with a search for a trade agreement list on other definitions.

Task 5.1.4. Specifying Additional Information for the Outputs

Remember that each trade agreement profile can have more than one output. For each output you can specify alternative IDs and locations. If you enter values on the **Sending Partner** tab of the Partner Relationship Trade Agreement Options window, the IDs can be placed in the output wrapper. The location becomes the source location of the outbound file.

The screenshot shows the 'Partner Relationship Trade Agreement Options: EXAMPLE-X850' dialog box. The 'Output' dropdown is set to '1,EXAMPLE,1,FPO' (1 of 1). The 'Sending Partner' tab is active. The 'Partner Name' is 'Example Output Sending Partner'. The 'Location' is empty, and the 'Override Location' is 'TESTSEND-MAILBOX'. The 'Interchange' section has 'ID' set to 'TESTSEND' and 'Qual' empty. The 'Functional Group' section has 'ID' empty and 'Qual' empty. 'Int ID' and 'Int ID2' fields are also empty. The 'OK', 'Cancel', and 'Apply' buttons are at the bottom.

Field	Value
Output	1,EXAMPLE,1,FPO (1 of 1)
Partner Name	Example Output Sending Partner
Location	
Override Location	TESTSEND-MAILBOX
Interchange ID	TESTSEND
Interchange Qual	
Interchange Int ID	
Interchange Int ID2	
Functional Group ID	
Functional Group Qual	
Functional Group Int ID	
Functional Group Int ID2	

If you enter values on the **Recipient Partner** tab of the Partner Relationship Trade Agreement Properties window, the IDs can be placed in the output wrapper, and the location is identified as the destination location of the outbound file.

The screenshot shows a dialog box titled "Partner Relationship Trade Agreement Options: EXAMPLE-X850". At the top, there is an "Output:" dropdown menu with the value "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner", "Recipient Partner" (which is selected), and "Misc".

Under the "Recipient Partner" tab, there are several input fields:

- "Partner Name:" with a dropdown menu showing "Example Output Recipient Partner".
- "Location:" with an empty text box.
- "Override Location:" with a text box containing "TESTREC-MAILBOX".

There are two sections for identifiers:

- Interchange:** Contains an "ID:" field with "TESTREC" (highlighted with a red box), a "Qual:" field, and "Int ID:" and "Int ID2:" fields.
- Functional Group:** Contains an "ID:" field, a "Qual:" field, and "Int ID:" and "Int ID2:" fields.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

You can do the following from the miscellaneous page:

- Set flags in the outbound wrapper to request a return acknowledgment (type TA1 for X12) and to specify if this is a test
- Specify alternative service characters for the outbound document, if this is a delimited standard
- Force this document to be placed in its own interchange by selecting **Document Level Break**.
- Identify whether an acknowledgment is expected for this document (used for the optional reconciliation process)

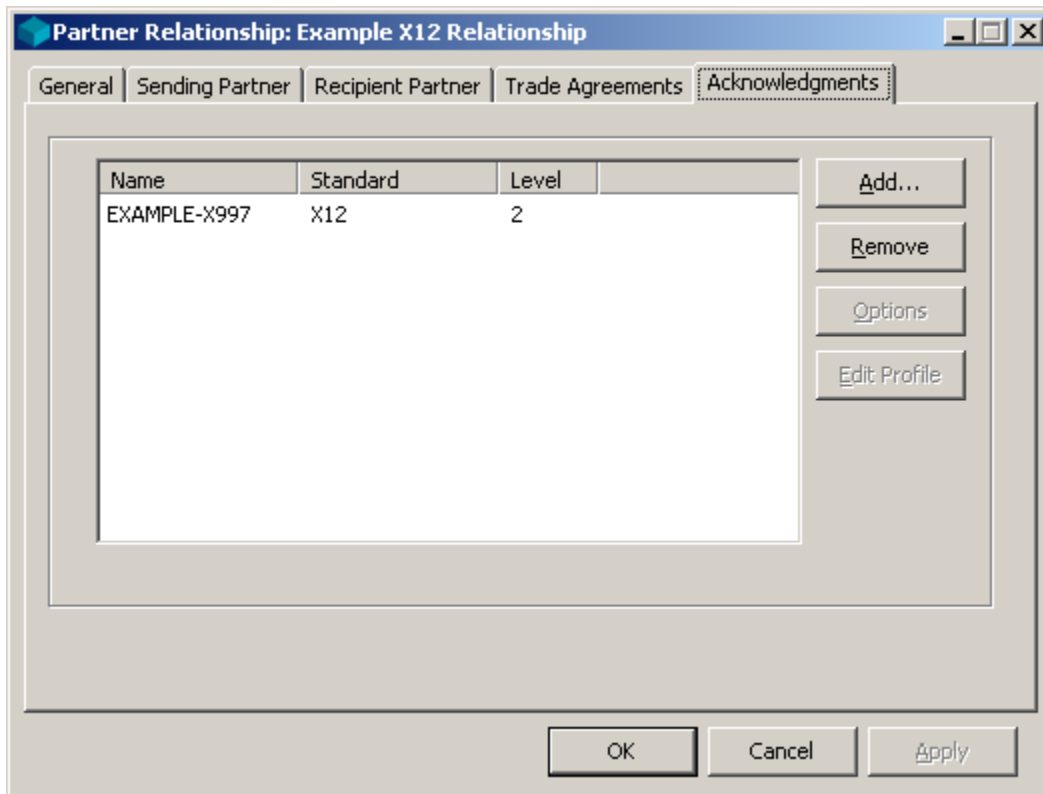
The screenshot shows a dialog box titled "Partner Trade Agreement Options: EXAMPLE-X850". At the top, there is a dropdown menu for "Output" set to "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner", "Recipient Partner", and "Misc", with "Misc" being the active tab. The "Misc" tab contains several sections:

- Output Fields:** Two checkboxes, "Request Acknowledgment" and "Test", both of which are currently unchecked.
- Service Characters:** A section with six input fields for "Segment Terminator", "Component Delimiter", "Release Character", "Tag Delimiter", "Element Delimiter", and "Repeat Separator". Each field has a small square icon to its right.
- Flags:** Three checkboxes: "Document Level Break", "Output Validation", and "Acknowledgment Expected", all of which are currently unchecked.

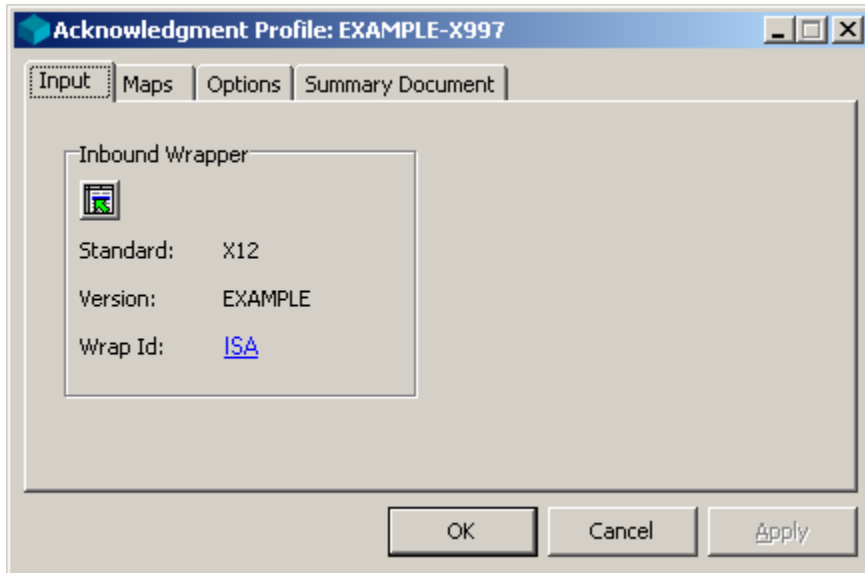
At the bottom of the dialog box are three buttons: "OK", "Cancel", and "Apply".

Task 5.1.5. Specifying Acknowledgment Profiles

If you want to return an acknowledgment to the sending partner, you do so by adding the acknowledgment profile to the list on the **Acknowledgments** tab.

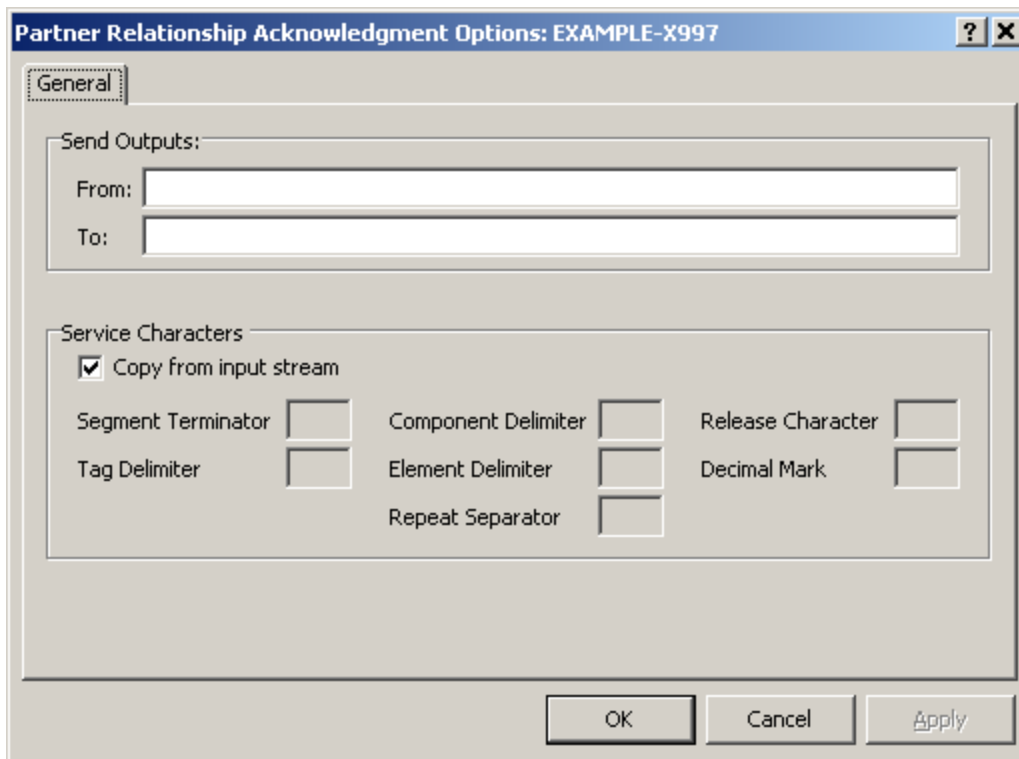


You can generate up to nine acknowledgments on the list. To be generated, they must match the inbound wrapper specified on the Input page of the Acknowledgment Profile window.



You can specify the following for the acknowledgment:

- Alternative location IDs for the acknowledgment in the **From** and **To** boxes
- Alternative service characters to generate a delimited acknowledgment






Task 5.1.6. Generating the Partner Relationship Text File

To generate the text file, select **Partner File** from the **Generate** menu. This places all partner, partner relationship, and group definitions in the **Partner.txt** file.

Task 5.1.7. Printing a Partner Relationship Report

To view a report of your partner relationship, select the partnership in the right pane.

From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file
- The **Print Preview** button  to print reports to the screen
- The **Print to PDF** button  to print to a PDF file

MW Translator Workbench Partner Relationship Report

RelationshipName: Example X12 Relationship

Sending Partner: Example X12 Sender

Location: X12-SEND-LOC

Level 1 Id: ICH-SEND-ID **Qual:** ZZ

Int Id:

Int Id 2:

Level 2 Id: FG-SEND-ID **Qual:**

Int Id:

Int Id 2:

Recipient Partner: Example X12 Recipient

Location: X12-REC-LOC

Level 1 Id: ICH-REC-ID **Qual:** ZZ

Int Id:

Int Id 2:

Level 2 Id: FG-REC-ID **Qual:**

Int Id:

Int Id 2:

Trade Agreements **Use Default TA** T

TA Profile Name: EXAMPLE-X850

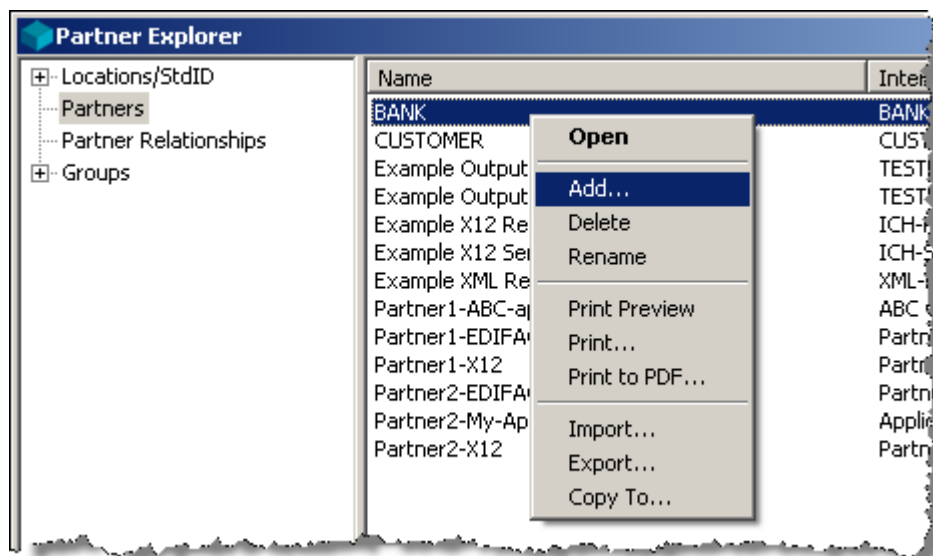
Output Seq 1 **Ack. Req** F **Req. Test** F

Task 5.2. Creating Individual Partners

You may want to create individual partner definitions, because you do not need to define the more restrictive partner relationship.

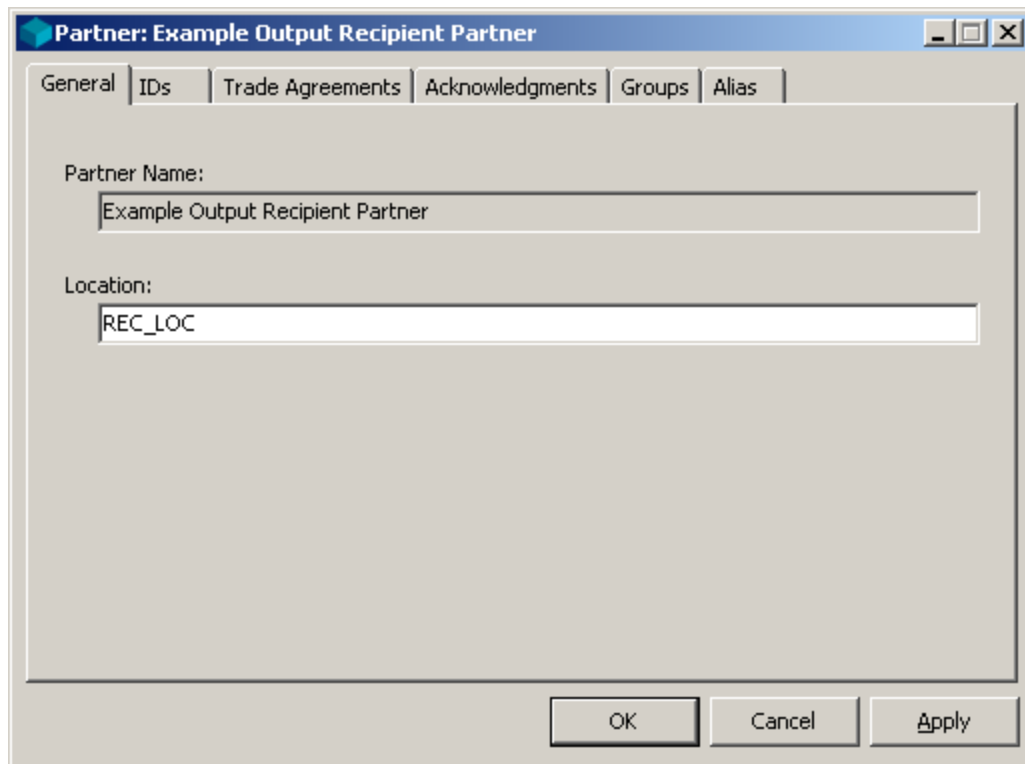
Task 5.2.1. Specifying the Partner Name

You can create a single partner by selecting the **Partners** folder then **Add** from the **File** menu or right-click and **Add** from the menu. Any partner you create can be either the sender or recipient.



Task 5.2.2. Specifying a Location

You enter the location for this partner on the **General** tab. When this partner is the sender, this value is the default source location of the output. When this partner is the recipient, this value is the default destination location of the output. This default value will be used in case no other location value is found.



The screenshot shows a dialog box titled "Partner: Example Output Recipient Partner". The dialog has a tabbed interface with the following tabs: "General", "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "General" tab is selected. Inside the dialog, there are two text input fields. The first is labeled "Partner Name:" and contains the text "Example Output Recipient Partner". The second is labeled "Location:" and contains the text "REC_LOC". At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Apply".

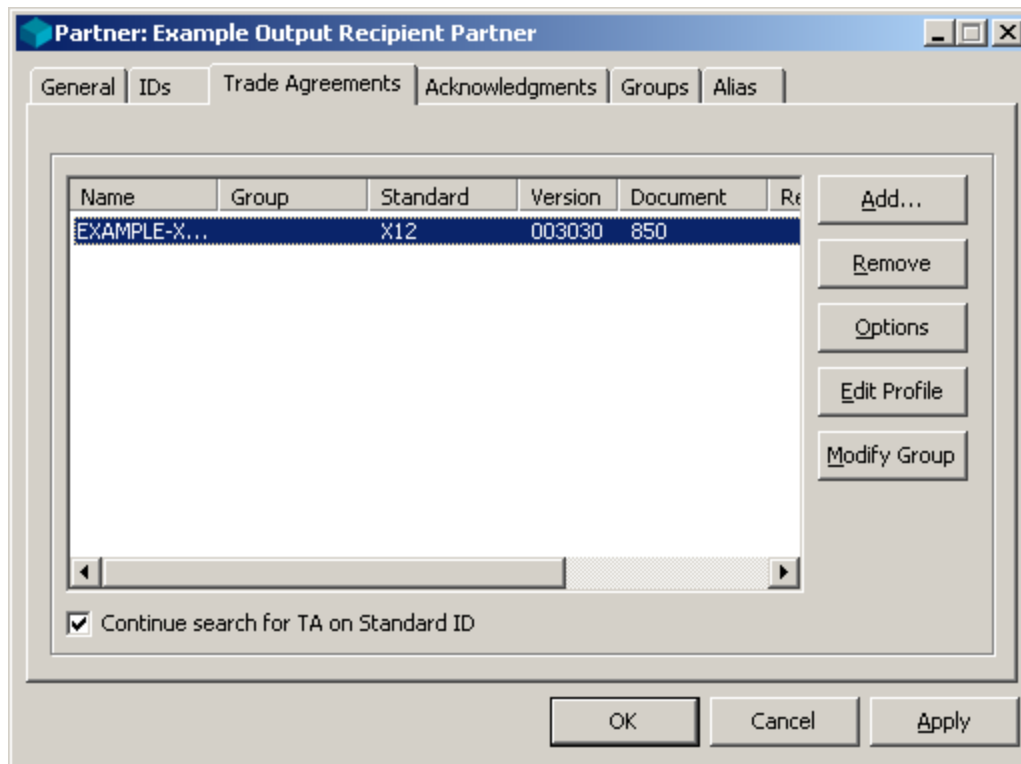
Task 5.2.3. Specifying Partner IDs

The partners may exist, but you now must specify IDs for the partner to match the incoming data.

The screenshot shows a dialog box titled "Partner: Example Output Recipient Partner" with a blue header bar. Below the title bar are several tabs: "General", "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "IDs" tab is currently selected. The dialog is divided into two main sections: "Interchange" and "Functional Group". Each section contains three input fields: "ID:", "Int ID:", and "Int ID2:". In the "Interchange" section, the "ID:" field contains the text "TESTREC" and the "Qual:" field is empty. The "Int ID:" and "Int ID2:" fields are also empty. The "Functional Group" section has empty fields for "ID:", "Int ID:", and "Int ID2:". Below these sections is a "More..." button. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Task 5.2.4. Specifying the Trade Agreement Profile

If this partner is to be a recipient, and you want to control processing based on this partner ID without concern about the sending partner, you need to associate the processing control instructions with this partner definition. You do this by adding a trade agreement profile to a list on the **Trade Agreements** tab.



An attempt is made to match each of the profiles with the inbound data, using the first profile that matches the **Input Field Values** on the Trade Agreement Profile itself. The standard identified during the initial phases of processing must match the value in the **Identified Standard** field.

The screenshot shows a dialog box titled "Trade Agreement Profile: Partner2-X850". It has three tabs: "Matching Fields", "Input", and "Outputs". The "Matching Fields" tab is active. On the left, under "Identified Standard:", there is a dropdown menu showing "X12". On the right, under "Input Field Values:", there are five text input fields: "Doc Id" with "850", "Version" with "003030", "Release" (empty), "Agency" with "X", and "Assoc" (empty). At the bottom, there are three buttons: "OK", "Cancel", and "Apply".

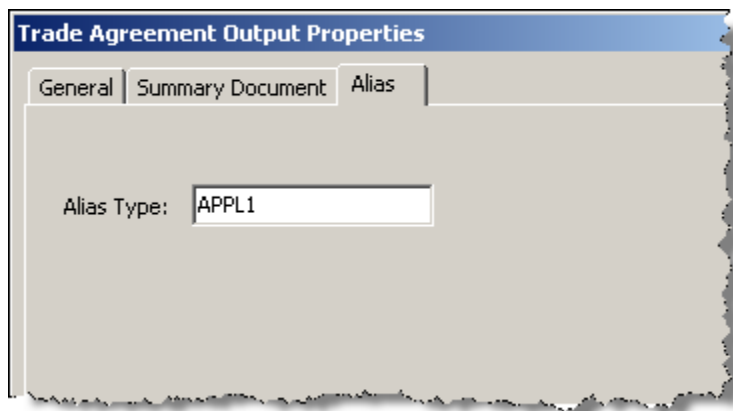
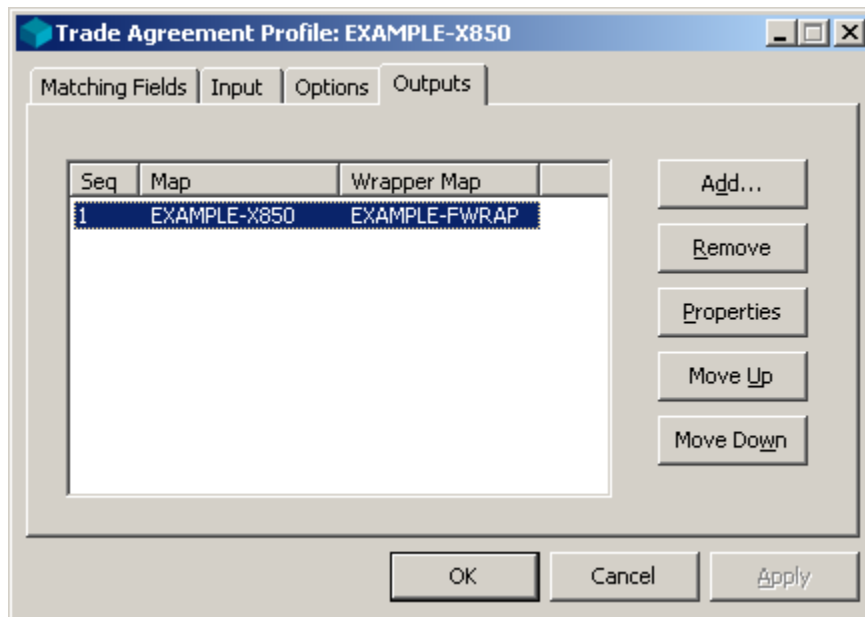
If there is no trade agreement on the list that matches the inbound data, and if you have checked **Continue search for TA on Recipient Partner**, processing continues with a search for a trade agreement list on other definitions.

Task 5.2.5. Specifying Alternative Information for the Outputs

Remember that each trade agreement profile can have more than one output. For each output you may specify alternative IDs and location. If you cannot predict the sender, you must use aliasing to determine an alternative output ID, and perhaps location. If you also want alternative IDs for the recipient in the output wrapper, it makes sense to also use aliasing for the recipient partner. Otherwise, you can enter alternative IDs directly on the **Recipient Partner** tab of the Partner Trade Agreement Options window.

For more information, refer to the topic, *Specifying Alternative Output IDs or Routing* (on page 257).

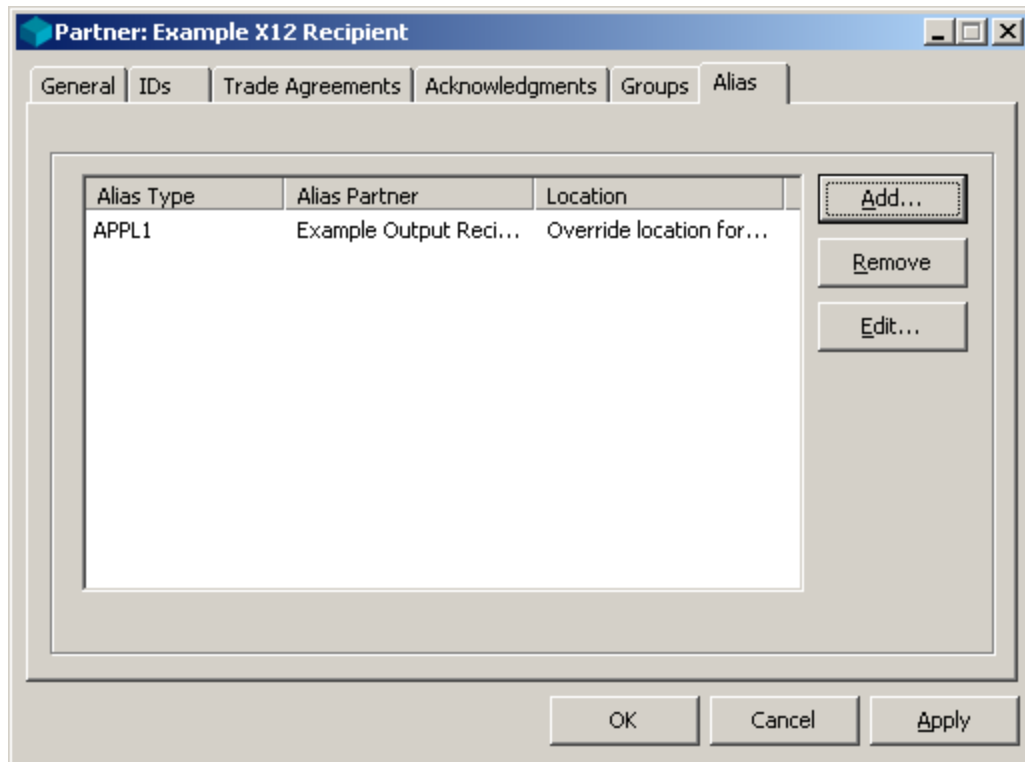
Let us assume you will use aliasing. You begin by associating the trade agreement output with an alias type. You do this from the Trade Agreement Outputs Properties window, accessible from the **Outputs** tab of the Trade Agreement Profile window.



Then you link your incoming partner with your aliased partner to be used for output.

NOTE: The aliased partner definition must already exist. If it does not, you must first create it.

You link the two partner definitions from the Alias page of the inbound partner, using the same alias type as you used for the trade agreement output properties.



You can do the following from the miscellaneous page:

- Set flags in the outbound wrapper to request an acknowledgment (TA1 type for X12) and to specify if this is a test
- Specify alternative service characters for the outbound document, if this is a delimited standard
- Force this document to be placed in its own interchange by selecting **Document Level Break**.
- Identify whether an acknowledgment is expected for this document for the optional reconciliation process
- Specify whether you want to validate the output (default is no validation on output)

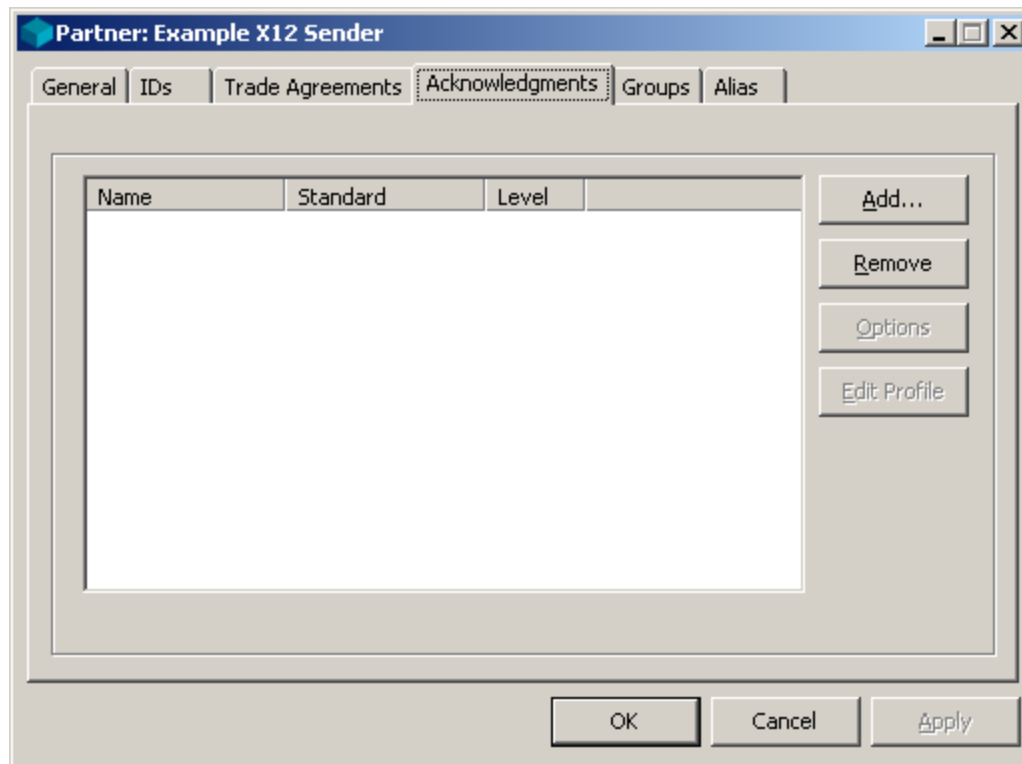
The screenshot shows a dialog box titled "Partner Trade Agreement Options: EXAMPLE-X850". At the top, there is a dropdown menu for "Output:" with the value "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner", "Recipient Partner", and "Misc", with "Misc" being the active tab. The dialog is divided into several sections:

- Output Fields:** Contains two checkboxes: "Request Acknowledgment" and "Test", both of which are currently unchecked.
- Service Characters:** A section containing six input fields for defining delimiters: "Segment Terminator", "Component Delimiter", "Release Character", "Tag Delimiter", "Element Delimiter", and "Decimal Mark". There is also a "Repeat Separator" field below the others. All these fields are currently empty.
- Flags:** A section containing three checkboxes: "Document Level Break", "Output Validation", and "Acknowledgment Expected". All three are currently unchecked.

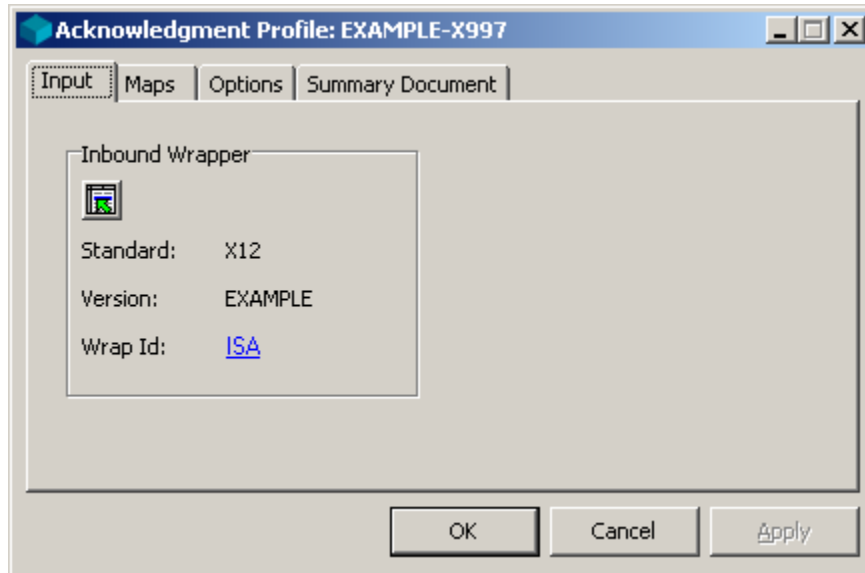
At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Task 5.2.6. Specifying Acknowledgment Profiles

When this partner is a sender, you can specify if you want to return acknowledgments. If you want to return an acknowledgment to this partner when the sender, you do so by adding the acknowledgment profile to the list on the **Acknowledgments** tab.

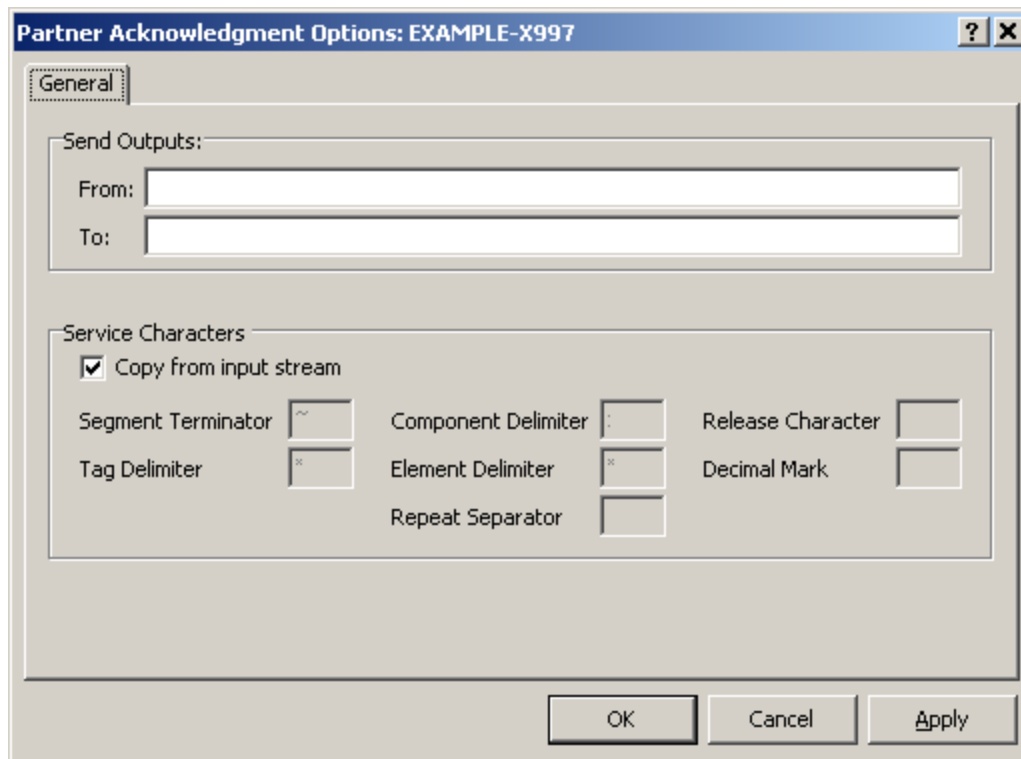


You can generate up to nine acknowledgments on the list. To be generated, they must match the inbound wrapper specified on the **Input** tab of the Acknowledgment Profile window.



You can specify the following for the acknowledgment:

- Alternative location IDs for the acknowledgment in the **From** and **To** boxes
- Alternative service characters to generate a delimited acknowledgment






Task 5.2.7. Generating the Partner Text File

To generate the text file, select **Partner File** from the **Generate** menu. This places all partner, partner relationship, and group definitions in the **Partner.txt** file.

Task 5.2.8. Printing a Partner Report

To view a report of your partner, select the partner in the right pane.

From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file
- The **Print Preview** button  to print reports to the screen
- The **Print to PDF** button  to print to a PDF file

MW Translator Workbench Partner Report

PartnerName: Example Output Recipient Partner

Location:

Level 1 Id: TESTREC

Int Id:

Int Id 2:

Qual:

Creating Definitions Using the Partner Wizard

If you are unfamiliar with how partners work and how to define them, it might be easier to create partner definitions using the Partner Wizard. The Wizard allows you to create new definitions only. To modify existing definitions, you must do so from Partner Explorer.

You can do the following using the Partner Wizard:

- Define a new partner relationship
- Define a new single partner
- Define a new inbound location and standard ID

Since we are discussing partner definitions here, we will discuss only the first two options.

You can do several things from Partner Explorer that you cannot do using the Partner Wizard as follows:

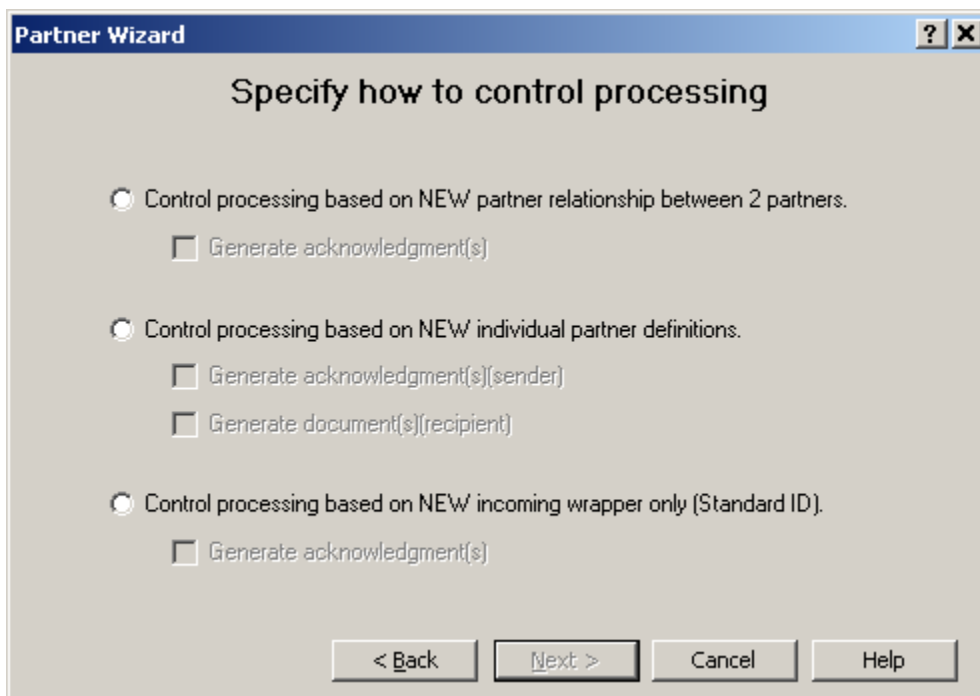
- Define closed groups
- Define aliases
- Change any existing definitions

IMPORTANT: You must already have defined any trade agreement profiles or acknowledgment profiles that you want to reference during this process.

Specify How to Control Processing

Which configurations you create depend on the selections you make regarding your decision on how to control processing. What you are deciding is whether and what type of matching must occur on partner IDs in order to find the appropriate trade agreement profile. Since we are discussing partners here, the assumption is that you want to use partner definitions to control processing. Remember that the trade agreement profile specifies what work is to be done and the definitions required to do the work.

For more information about the implications of each choice, refer to the section, *Determining Processing Control* (on page 17).



When you tell the Partner Wizard how you want to control processing, it will look for and use or create new definitions as required. It will assume that you want to perform certain matching tasks, which will give you a better idea of the difference between the choices. Here is what each of the choices means.

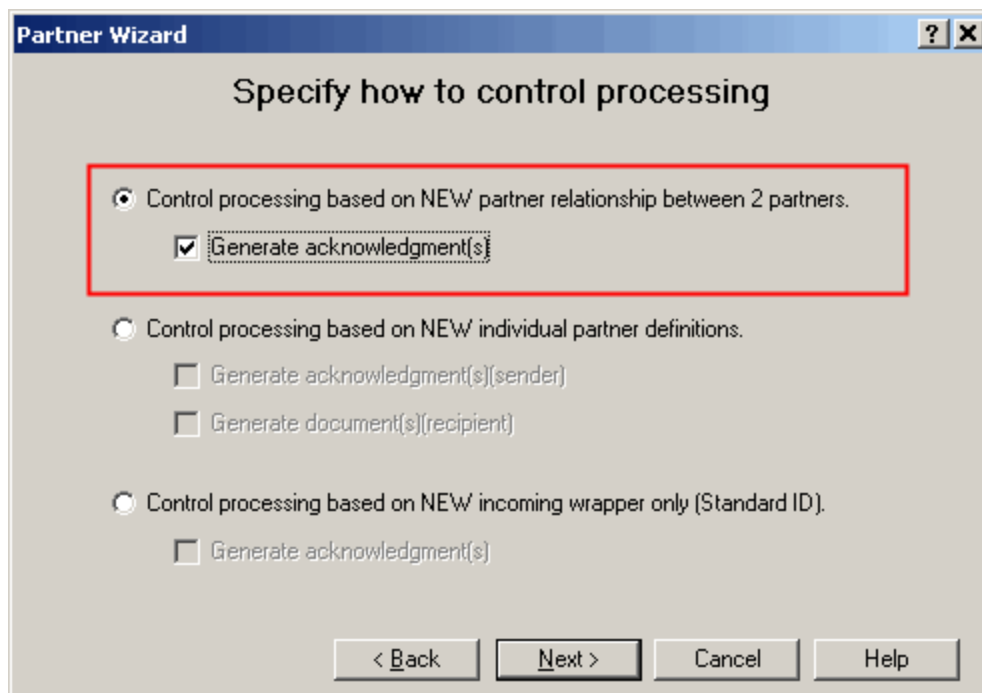
Type of Control	Partner folder	Partner Matching Tasks
Partner relationship between 2 partners	Partner Relationships	Requires sending and recipient partner IDs
Individual partner definitions	Partners	Requires inbound partner ID only

Once you make a choice, the Partner Wizard presents a series of appropriate windows to lead you through the definitions tasks. You specify definitions in the following general order, though you may not have to or choose not to enter information for all of them.

TIP: Do not worry about making mistakes. You can always back out of any definitions you have entered by using the **Back** or **Cancel** buttons.

Control Processing Based on New Partner Relationship

When you select this option, you will always be asked to enter a trade agreement profile, but you can check the box to be asked to enter an acknowledgment profile. If you do not want to generate acknowledgments, you should not check this box.



The screenshot shows a dialog box titled "Partner Wizard" with a subtitle "Specify how to control processing". The dialog box contains three radio button options for controlling processing:

- Control processing based on NEW partner relationship between 2 partners.
 - Generate acknowledgment(s)
- Control processing based on NEW individual partner definitions.
 - Generate acknowledgment(s)[sender]
 - Generate document(s)[recipient]
- Control processing based on NEW incoming wrapper only (Standard ID).
 - Generate acknowledgment(s)

At the bottom of the dialog box, there are four buttons: "< Back", "Next >", "Cancel", and "Help".

When you select certain definitions, the Partner Wizard will require that you select predefined entities or, in some cases, allow you to create new definitions. These are as follows when you want to control processing based on a partner relationship:

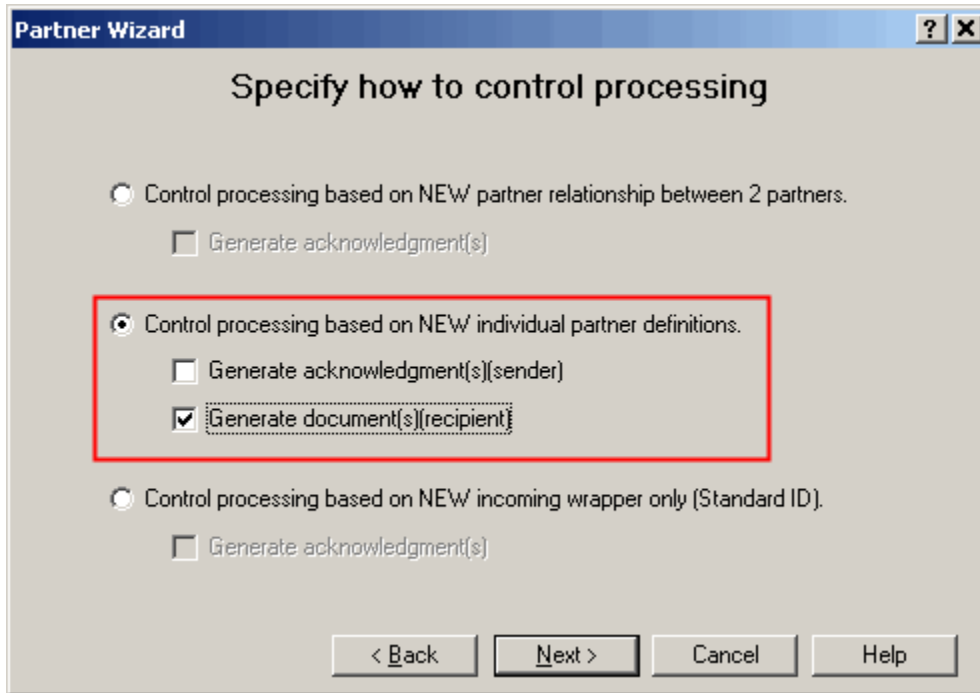
Definition	Description and Purpose
Partner Relationship and Sending and Recipient Partners	You create a new partner relationship by defining everything as you proceed or by selecting an existing one as a template, which you then must change. The combination of partner IDs must be unique for all relationships. If the partners do not exist they will be created. You specify IDs to match the inbound data and to be used as default IDs for the output wrappers. The locations will be used as default source and destination locations.
Trade Agreement Profile	You must select an existing trade agreement profile. There is little reason to create a partner relationship without specifying a trade agreement profile. The profile specifies the work that must be done and the definitions required to do the work.
Trade Agreement Output Options (appears when you select a trade agreement profile)	This is one way to specify alternative IDs for the output and source and destination locations, if they must be different from what you have specified for the input partner definitions.
Acknowledgment Profile (only appears if you check the Generate Acknowledgment(s) box)	If you must generate an acknowledgment for the sending partner, you must select an existing acknowledgment profile.
Acknowledgment Options (appears when you select an acknowledgment profile)	Specify overriding source and destination locations and, if applicable, service characters for a delimited acknowledgment

Control Processing Based on New Individual Partner Definitions

When you select this option, you may or may not want to specify a trade agreement profile or an acknowledgment profile, or may want to specify both. If you do not want to specify an acknowledgment, leave the **Generate acknowledgment(s)** box unchecked. If you do not want to specify a trade agreement, leave the **Generate document(s)** box unchecked. The following table shows you the combinations of configurations you can create and the typical purpose for doing so:

Configuration Options	Purpose
Partner, without generating either an acknowledgment or an output document.	Use partner IDs to match with inbound data for security reasons Use partner IDs as alternative IDs for outbound wrappers or location as alternative location
Partner, and generate outbound document(s)	Specify when and how to generate data for this recipient partner by selecting trade agreement profile
Partner, and generate acknowledgment(s)	Specify when and how to generate acknowledgment(s) for this sending partner by selecting acknowledgment profile
Partner, and generate both outbound data and acknowledgment(s) depending on role of partner as sender or recipient	Specify when and how to generate data when this partner is the recipient (trade agreement profile), and Specify when and how to generate acknowledgment(s) when this partner is the sender (acknowledgment profile).

A common option is to define a recipient partner for which you want to generate output.



When you select certain definitions, the Partner Wizard will require that you select predefined entities or, in some cases, allow you to create new definitions. These are as follows when you want to control processing based on individual partner definitions:

Definition	Description and Purpose
Partners	You create a new partner by defining everything as you proceed or by selecting an existing one as a template, which you then must change. The combination of partner IDs must be unique for all partners. If the partner does not exist it will be created. You specify IDs to match the inbound data and to be used as default IDs for the output wrappers. The location will be used as the default source or destination location depending on whether the partner is the sender or recipient.
Trade Agreement Profile (only appears when you check the Generate Document(s) (recipient) box)	You must select an existing trade agreement profile. The profile specifies the work that must be done and the definitions required to do the work.
Trade Agreement Output Options (appears when you select a trade agreement profile)	This is one way to specify alternative IDs for the output and source and destination locations, if they must be different from what you have specified for the input partner definitions.

Definition	Description and Purpose
Acknowledgment Profile (only appears if you check the Generate Acknowledgment(s) box)	If you must generate an acknowledgment for this partner as the sending partner, you must select an existing acknowledgment profile.
Acknowledgment Options (appears when you select an acknowledgment profile)	Specify overriding source and destination locations and, if applicable, service characters for a delimited acknowledgment

Configuring Standard ID and Defaults

Overview

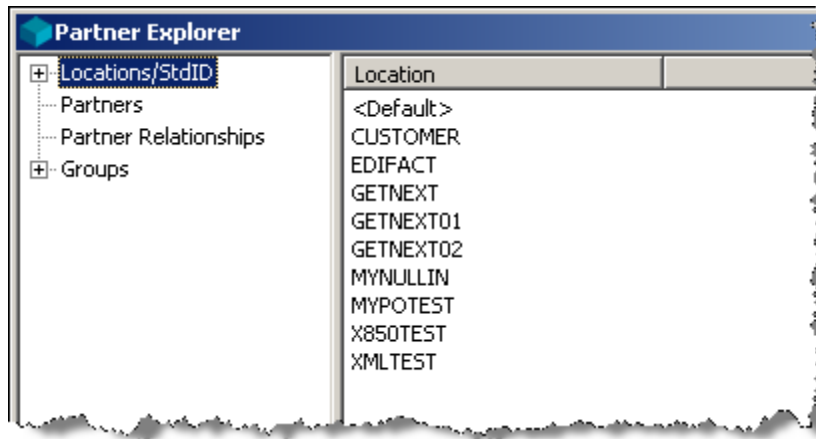
The first mandatory processing task is to identify the incoming wrapper. Here you associate the incoming wrapper definition with a location, either a specific one or <Default>.

Standard identification information has two basic purposes:

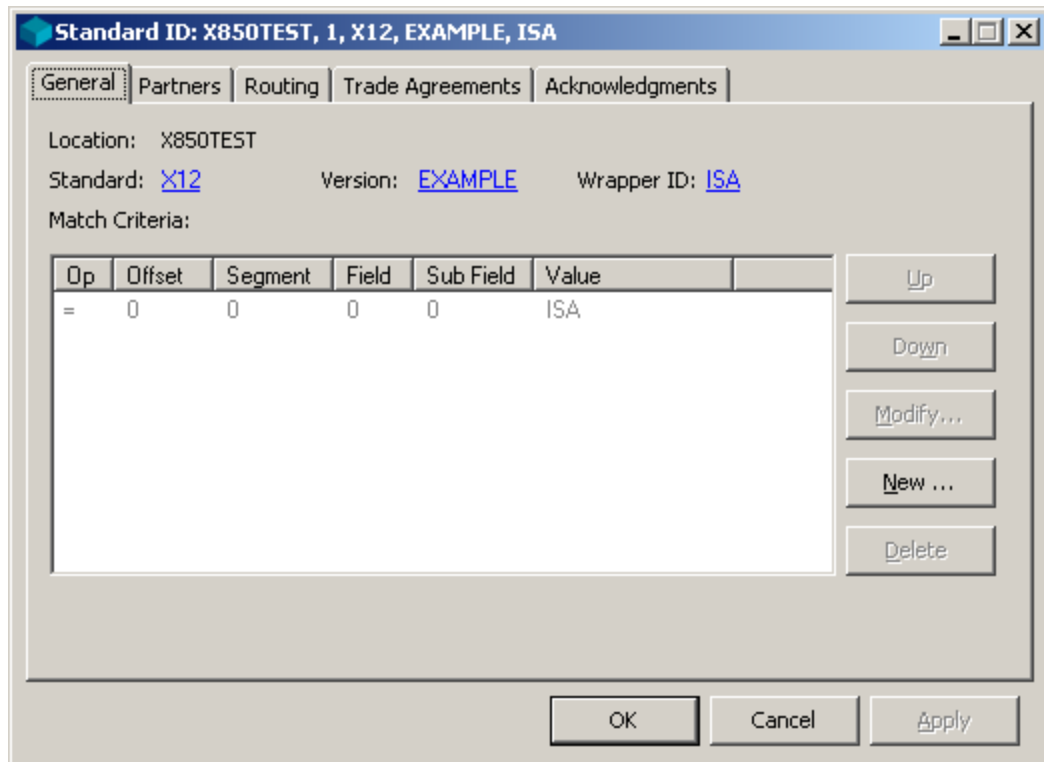
- Specify matching criteria to select the inbound wrapper definition required to parse the wrapper information
- Specify default processing instructions for routing, partners, trade agreements, and acknowledgments

Using Partner Explorer or the Partner Wizard

You access definitions related to standard identification from the **Locations/StdID** folder of Partner Explorer, which displays the list of input locations you have defined in the left pane and the specific wrappers within each location in the right pane.



Tabbed pages organize the definitions for standard identification.



Default routing information is organized separately by definitions for output and acknowledgment.

Standard ID: X850TEST, 1, X12, EXAMPLE, ISA

General Partners **Routing** Trade Agreements Acknowledgments

Location: X850TEST
Standard: X12 Version: EXAMPLE Wrapper ID: ISA

Default Output Routing

Data Field: Partner ID Source Address Use Source before Data Field
 Partner Int ID Use Source before Partner Location

Default: Source:
Destination:

Default Acknowledgment Routing

Data Field: Partner ID Source Address Use Source before Data Field
 Partner Int ID Use Source before Partner Location

Default: Source:
Destination:

OK Cancel Apply

Using the Partner Wizard

The Partner Wizard will lead you through the maze of setting up new partnerships and locations and standard IDs. If you need to modify definitions, you must do so directly from Partner Explorer.

Once you have been led through the definitions process, the wizard displays a list of definitions that it will use for the new partnership for you to review.

For a discussion of using the Partner Wizard to create standard ID definitions, refer to the topic *Creating Standard ID Definitions Using the Partner Wizard* (on page 313).

Task 6. Specifying the Source Location

From a list of wrappers, the TRM must choose the correct one to parse the incoming wrapper data. It does this by first looking in the source location. If the source location does not exist, it searches the <Default> location. If it cannot find a matching wrapper, the TRM terminates processing.

MessageWay will pass a source location to MW Translator differently, depending on the adapter or service that delivers the message to MW Translator. The source might come from any of the following:

- MessageWay configuration
- MW Translator configuration
- E-mail server
- Status file in a script

The following table shows how MessageWay determines the name of the source location for MW Translator (the adapters or services that are not part of base MessageWay are described as options).

Input Adapter or Server	Source	Notes
AS2 Server (option)	AS2 server AS2-From address on the AS2 message.	Default For messages that are signed, this value uniquely matches part of the sender's private key subject.
Custom IO Adapter	MessageWay (Name of Custom IO input site)	Default
	Text file (Script that contains status file with optional name of source location)	Overrides default location
Custom Processing Service	MessageWay (Name of original input location that sent message to custom processing service location)	Default
	Text file (Script that contains status file with name of source location)	Overrides default location
Disk Transfer Adapter	MessageWay (Name of Disk Transfer input site)	Default



Input Adapter or Server	Source	Notes
	MessageWay (Site Properties window Disk Input tab Sender box)	Overrides default location
Distribution List Adapter	MessageWay (Name of original input location that sent message to distribution list service location)	Default
	MessageWay (Service Location Properties window, Distribution List Recipients box, Sender (opt.) column)	Overrides default location
E-mail Adapter	E-mail server (Name of sender's e-mail account and e-mail address passed from the e-mail server)	Default
	MessageWay (Site Properties window POP3 tab Sender box)	Overrides default location
FTP Adapter	MessageWay (Name of FTP input location)	Default
	MessageWay (Site Properties window FTP Input tab Sender box)	Overrides default location
FTP Server (option)	MessageWay (User Properties window Locations tab, Default Location box)	Default
MQ Adapter (options)	MessageWay (Name of MQ input location)	Default
	MessageWay (Site Properties window MQ Input tab Sender box)	Overrides default location

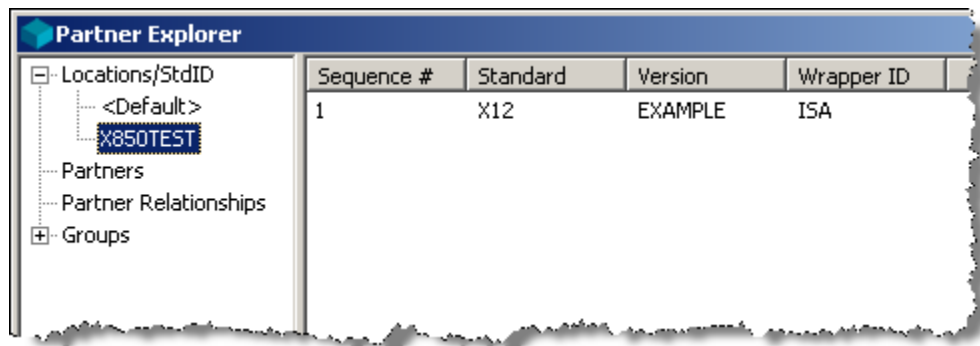
Input Adapter or Server	Source	Notes
Rules Processing	MessageWay (Name of original input location that sent message to rules service location)	Default
	MessageWay (Process Rule window, Action tab, Sender box)	Overrides default location
SFTP Adapter (option)	MessageWay (Name of FTP input location)	Default
	MessageWay (Site Properties window SFTP Input tab Sender box)	Overrides default location
SFTP Server (option)	MessageWay (User Properties window Locations tab, Default Location box)	Default
Translation Service (option)	MWTranslator (Sender's location, which is source location of output as determined by MWTranslator configurations)	Used when translation output is routed back to MWTranslator

Task 6.1. Creating the Source Location

You can use a specific source location to enter your standard identification information or you can use the <Default> location. If a standard is only received from one or a small number of locations, then enter those definitions within separate locations; otherwise use <Default>. For more information about how this source location is used within some messaging systems, refer to the online help for the **General** tab on the Standard Identification window.

If you want to create a new location, you add the location name by selecting the **Locations/StdID** folder and then **Add** from the **File** menu. Then you add the wrapper to the location by selecting the new location within the **Locations/StdID** folder and then **Add** from the **File** menu.

The TRM will attempt to match the wrappers on the list in the order they appear. To change the location in the list, use the **Up** button  or the **Down**  button from the toolbar.



Task 6.2. Specifying How to Match the Wrapper

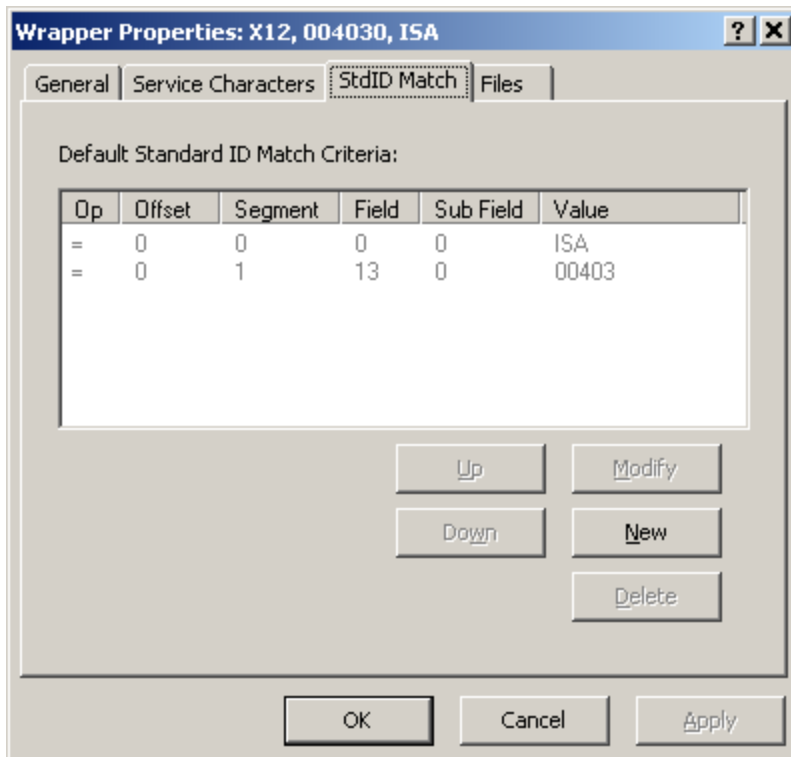
There are two methods to provide matching criteria for Standard ID:

- Automatically, by adding the matching criteria to the wrapper definition (Wrapper Properties window), which is entered when the wrapper is selected for standard identification
- Manually, by entering the matching criteria directly on the Standard ID window.

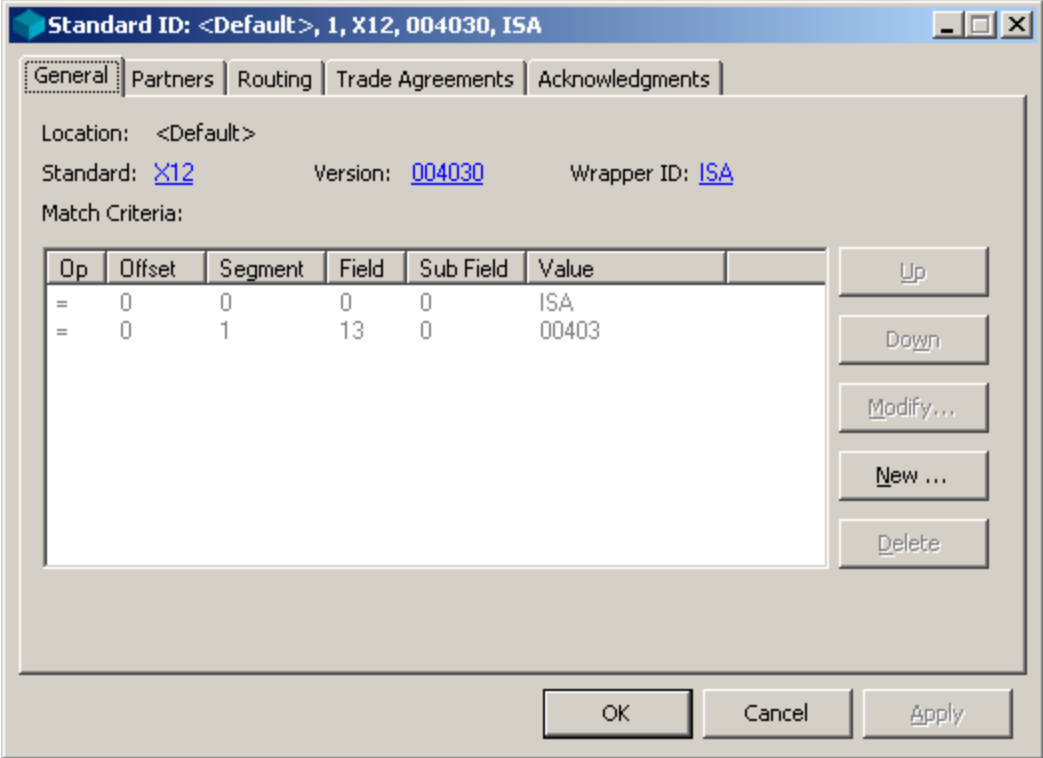
Users may enter matching values for a standard on the standard definition. We have already provided such matching criteria for X12 standards, and users may change or enter matching criteria for any standard definition.

When you add a new wrapper to a location in Partner Explorer for the standard identification process, these default values are automatically entered as matching criteria on the **General** tab of the Standard ID window.

For example, look at the **StdID Match** tab of the Wrapper Properties window for the ISA wrapper, X12 004030 standard version. Although the standard is delivered with these default values entered, you could enter default values for your proprietary standard by using the buttons on the window.



Then when you add a wrapper to a location, the default criteria appear as matching criteria on the **General** tab, as you can see here.



Users may enter values manually using one of two methods:

- Specify offset and value, or
- Specify segment, field, optionally sub field, and value.

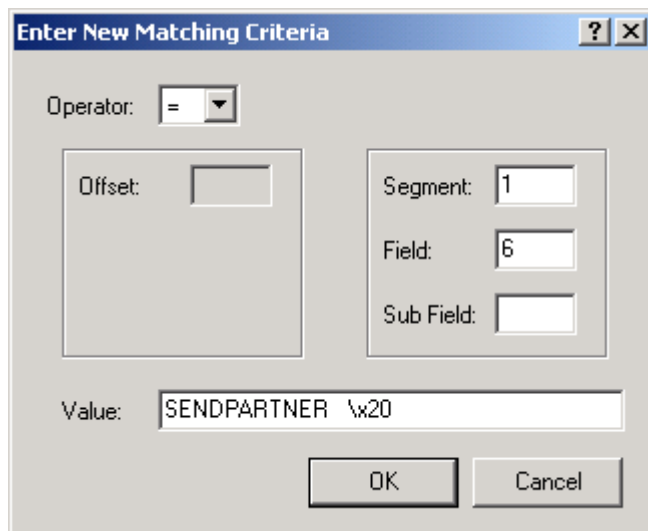
IMPORTANT: Once the TRM determines the definition of the first incoming wrapper from the standard identification process, it re-opens the input based on the standard and wrapper definitions. When the **IO Mode** is set to **Text**, the TRM replaces the carriage-return/linefeed (CRLF), linefeed (LF) or record break with a single new-line (NL) character for this and any subsequent interchanges in this file. For **IO Mode**, the TRM only uses the setting for the first interchange wrapper in the file. Therefore, when you have multiple interchanges in a file where the IO Mode is defined as Text on the first interchange wrapper, and when the end-of-segment marker is CRLF, and when you use offset values to identify matching criteria beyond the first segment, you must be sure that the offset for the matching criteria for subsequent interchanges allows for the change in segment terminators from two (CRLF) to one (NL) character.

You may enter hex values for non-printing characters in the format, $\backslash nn$, where nn is the hex value. For example, you may enter $\backslash x20$ for a space.

You must also enter an operator, which can be one of the following:

Operator	Function
=	equal
<>	not equal
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal

IMPORTANT: Trailing spaces entered for the value are deleted. To maintain trailing spaces to match an input value with trailing spaces, you may use a hexadecimal value at the end of the value field. For example, for an element of 15 characters, where the input value is **SENDPARTNER** followed by 4 spaces, in the value field of the **Enter New Matching Criteria** dialog box you would enter **SENDPARTNER \x20**, which is **SENDPARTNER** followed by three spaces terminated with a hexadecimal space.



The screenshot shows a dialog box titled "Enter New Matching Criteria". It features a title bar with a question mark and a close button. The main area contains several input fields: "Operator" is a dropdown menu showing "="; "Offset" is an empty text box; "Segment" is a text box containing "1"; "Field" is a text box containing "6"; and "Sub Field" is an empty text box. Below these is a "Value" text box containing "SENDPARTNER \x20". At the bottom of the dialog are "OK" and "Cancel" buttons.

The TRM will first attempt to match the values on the **Match Criteria** list in the order they appear, if any exist. You can change their order using the **Up** and **Down** buttons on the **General** tab.

IMPORTANT: If no values exist in the **Match Criteria** box, the wrapper matches automatically.

Task 7. Specifying Processing Requirements

You can specify whether to find partner definitions to process the data, and to override partner-based routing by using the source location instead.

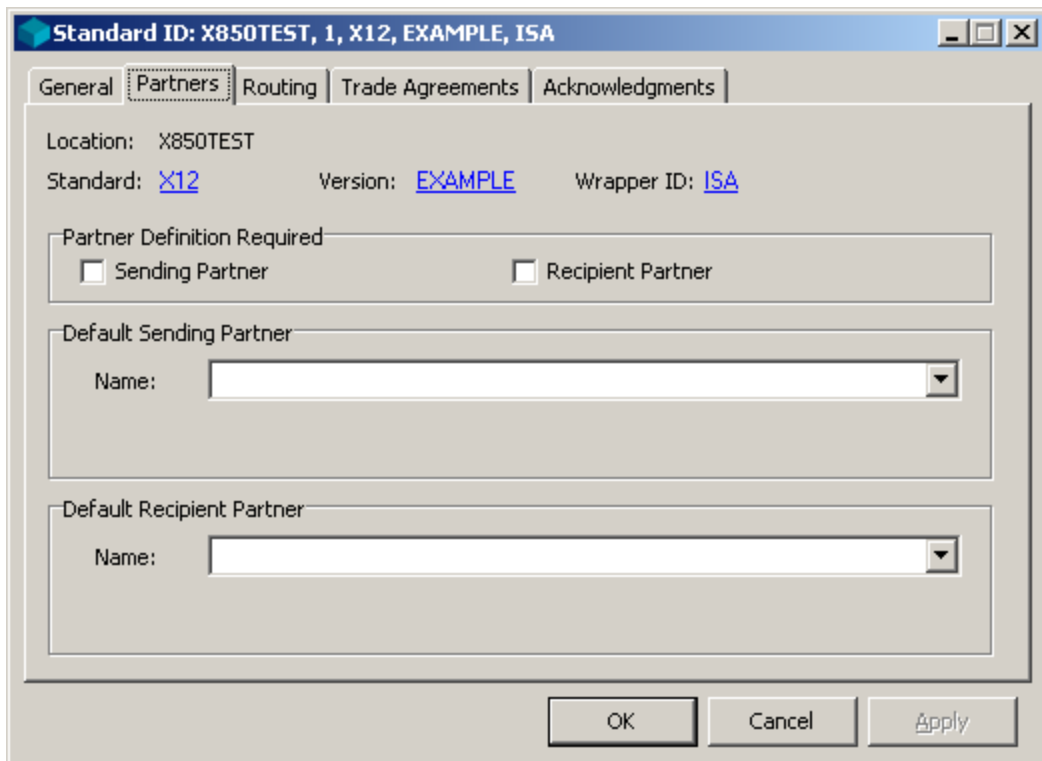
You can also specify default partner profiles for the sending or recipient partner if it cannot find them otherwise, default routing instructions, a default list of trade agreements, and a default list of acknowledgments.

Task 7.1. Specifying Processing Options for Partner Definitions

You can limit the work required by the TRM to find a trade agreement profile by specifying that it must match the sending or recipient partner with existing partner definitions. If it does not find the matching partner definitions, the TRM terminates processing.

You can also specify default partners to use in case the TRM cannot find any by matching partner IDs.

You enter these options on the **Partners** tab of the Standard ID window.



The screenshot shows a window titled "Standard ID: X850TEST, 1, X12, EXAMPLE, ISA". The "Partners" tab is selected. The window contains the following fields and controls:

- Location: X850TEST
- Standard: X12
- Version: EXAMPLE
- Wrapper ID: ISA
- Partner Definition Required:
 - Sending Partner
 - Recipient Partner
- Default Sending Partner:
 - Name: [Dropdown menu]
- Default Recipient Partner:
 - Name: [Dropdown menu]

At the bottom of the window are three buttons: OK, Cancel, and Apply.

Task 7.2. Specifying Default Routing Options

The TRM must find both a source and destination location for the output or it terminates processing. If it cannot find a location to route either the output data or a backward acknowledgment, you can provide a default strategy on the **Routing** page. The TRM will execute the selected options left to right and top to bottom, unless the order is modified by the option, **Use Source before Data Field**.

NOTE: The option **Use Source before Partner Location** is actually an override, not a default. This will force the TRM to use the source location used for the incoming wrapper during standard identification as the destination rather than any addresses it may have found in previous processing.

Standard ID: X850TEST, 1, X12, EXAMPLE, ISA

General | Partners | **Routing** | Trade Agreements | Acknowledgments

Location: X850TEST
Standard: [X12](#) Version: [EXAMPLE](#) Wrapper ID: [ISA](#)

Default Output Routing

Data Field: Partner ID Source Address Use Source before Data Field
 Partner Int ID Use Source before Partner Location

Default: Source:
Destination:

Default Acknowledgment Routing

Data Field: Partner ID Source Address Use Source before Data Field
 Partner Int ID Use Source before Partner Location

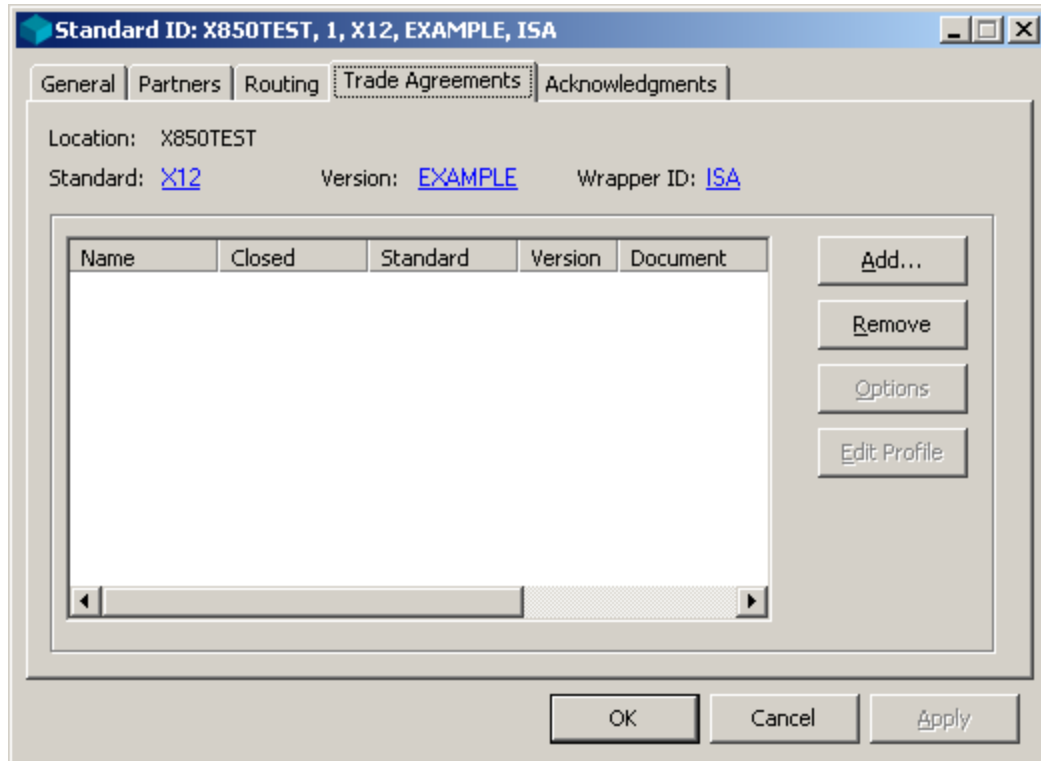
Default: Source:
Destination:

OK Cancel Apply

Task 7.3. Specifying Default Trade Agreements

You can specify a list of default trade agreement profiles in case the TRM is unable to find one. You must do this if you are not using partner definitions, or the TRM will terminate processing.

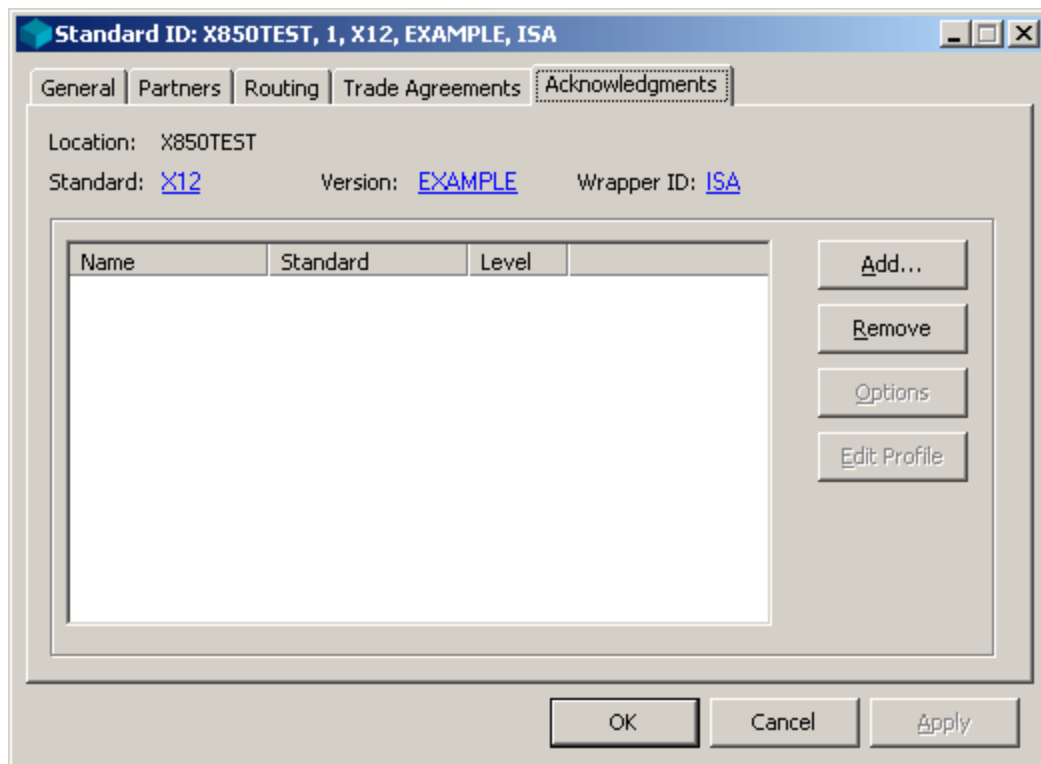
You specify these default trade agreement profiles on the **Trade Agreements** tab of the Standard ID window.



Task 7.4. Specifying Default Acknowledgments

You can specify a list of default acknowledgment profiles in case the TRM is unable to find any.

You specify these default acknowledgment profiles on the **Acknowledgments** tab of the Standard ID window.






Task 7.5. Generating the Standard ID Text File

You generate the standard ID text file by selecting **StdID File** from the **Generate** menu.

Task 7.6. Printing a Standard Identification Report

To view a Standard Identification report, select your standard ID wrapper in the right pane. From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file
- The **Print Preview** button  to print reports to the screen
- The **Print to PDF** button  to print to a PDF file

MW Translator Workbench Standard Id Report

Location: X850TEST

Standard ID: X12	EXAMPLE	ISA			
Sending Partner Req:		Recipient Partner Req:			
Default Sending Partner:					
Default Recipient Partner:					
Default Output Routing					
From Data Field	(invalid)	Use Source Before Data Field			
From Source Address		Use Source Before Partner Location			
Default					
Default Source					
Default Destination					
Default Acknowledgment Routing					
From Data Field	(invalid)	Use Source Before Data Field			
From Source Address		Use Source Before Partner Location			
Default					
Default Source					
Default Destination					
Match Criteria:					
Op	Offset	Segment	Field	Sub Field	Value
=	0	0	0	0	ISA

Creating Standard ID Using the Partner Wizard

If you are unfamiliar with how partners work and how to define them, it might be easier to create partner definitions using the Partner Wizard. The Wizard allows you to create new definitions only. To modify existing definitions, you must do so from Partner Explorer.

You can do the following using the Partner Wizard:

- Define a new partner relationship
- Define a new single partner
- Define a new inbound location and standard ID

Since we are discussing standard ID definitions here, we will discuss only last option.

You can do several things from Partner Explorer that you cannot do using the Partner Wizard as follows:

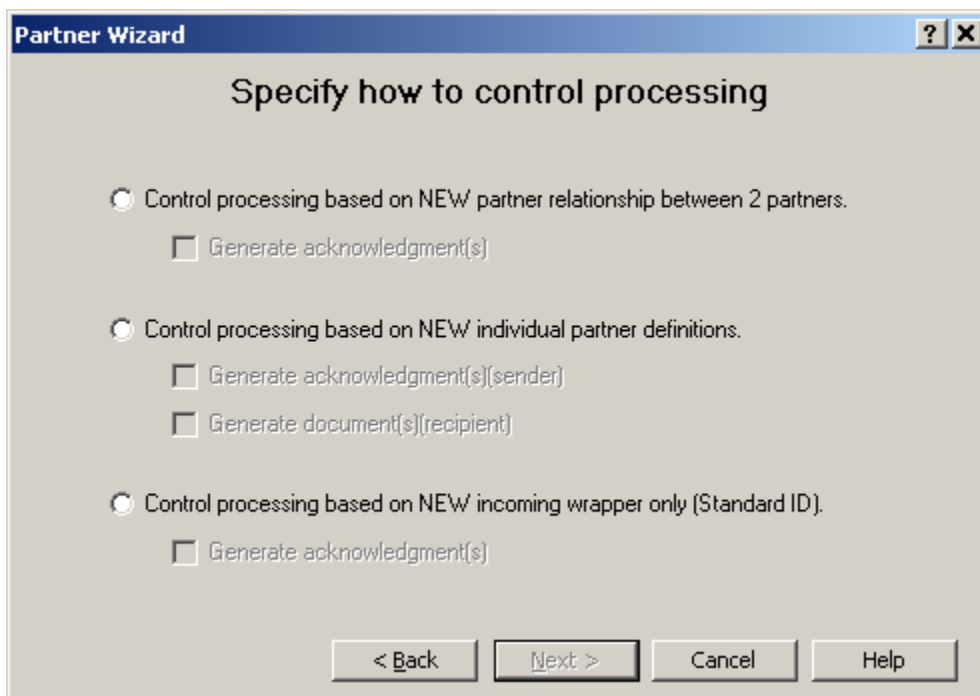
- Define closed groups
- Define aliases
- Change any existing definitions

IMPORTANT: You must already have defined any trade agreement profiles or acknowledgment profiles that you want to reference during this process.

Specify How to Control Processing

Which configurations you create depend on the selections you make regarding your decision on how to control processing. What you are deciding is whether and what type of matching must occur on partner IDs in order to find the appropriate trade agreement profile. Since we are discussing standard ID here, the assumption is that you want to use standard ID definitions to control processing. Remember that the trade agreement profile specifies what work is to be done and the definitions required to do the work.

For more information about the implications of each choice, refer to the chapter, *Determining Processing Control* (on page 17).



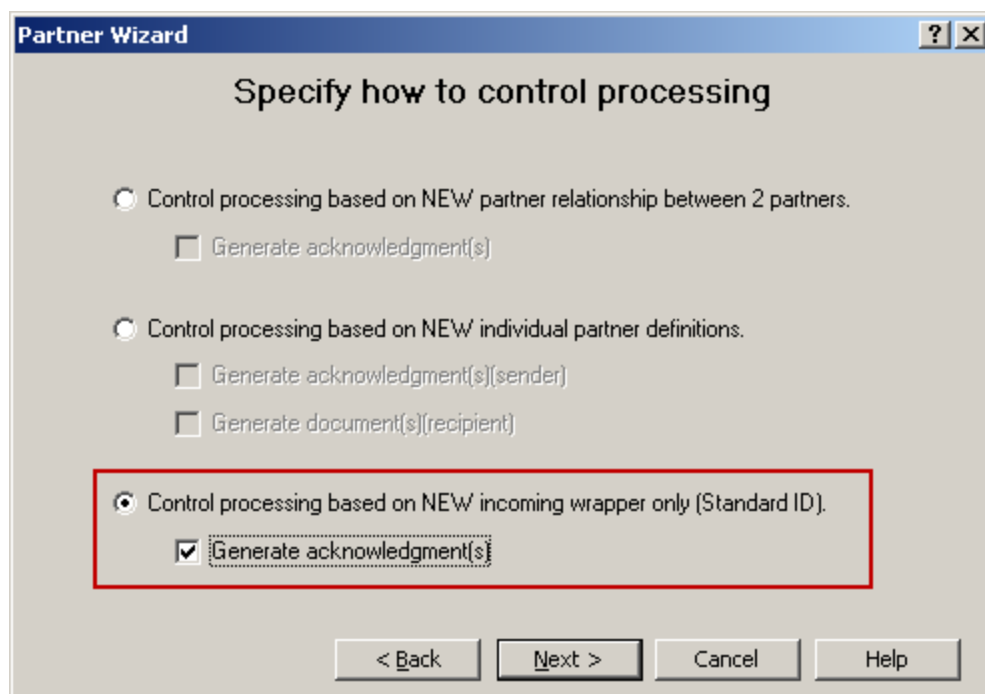
When you tell the Partner Wizard how you want to control processing, it will look for and use or create new definitions as required. It will assume that you want to perform certain matching tasks, which will give you a better idea of the difference between the choices.

Once you make a choice, the Partner Wizard presents a series of appropriate windows to lead you through the definitions tasks. You specify definitions in the following general order, though you may not have to or choose not to enter information for all of them.

TIP: Do not worry about making mistakes. You can always back out of any definitions you have entered by using the **Back** or **Cancel** buttons.

Control Processing Based on New Incoming Wrapper Only

When you select this option, you will always be asked to enter a trade agreement profile, but you can check the box to be asked to enter an acknowledgment profile. If you do not want to generate acknowledgments, you should not check this box.



When you select certain definitions, the Partner Wizard will require that you select predefined entities or, in some cases, allow you to create new definitions. These are as follows when you want to control processing based on a partner relationship:

Definition	Description and Purpose
Inbound (source) location	You create a new location by defining everything as you proceed or by selecting an existing one from the drop-down list. This is the location where the TRM will look for matching wrappers.
Inbound wrapper standard	You must select an existing wrapper definition to be associated with the location.
Trade Agreement Profile	You may select an existing trade agreement profile, if you want to control processing based on the identified standard, rather than on partner definitions. The profile specifies the work that must be done and the definitions required to do the work.

Definition	Description and Purpose
Trade Agreement Output Options (appears when you select a trade agreement profile)	This is one way to specify alternative IDs for the output and source and destination locations, if they must be different from what you have specified for the input partner definitions.
Acknowledgment Profile (only appears if you check the Generate Acknowledgment(s) box)	If you must generate an acknowledgment for anyone who sends you the identified wrapper, you may select an existing acknowledgment profile.
Acknowledgment Options (appears when you select an acknowledgment profile)	Specify overriding source and destination locations and, if applicable, service characters for a delimited acknowledgment

Special Cases. Standard ID Configurations

The following discussions provide information about how to configure standard ID definitions using various techniques to fulfill special requirements.

Using the Partner Wizard to Create a Standard ID

You can use the Partner Wizard to create a standard ID record to which you then link a trade agreement profile. In this case, the trade agreement profile receives an acknowledgment. Since acknowledgments are typically generic and do not require any special processing, you want to process the inbound acknowledgments as generically as possible. Thus, you associate the trade agreement profile with a standard ID wrapper definition, rather than a partner definition.

To Link the Trade Agreement for an Acknowledgment to a Standard ID

We are going to use the Partner Wizard to create our standard ID record, because it is easy.

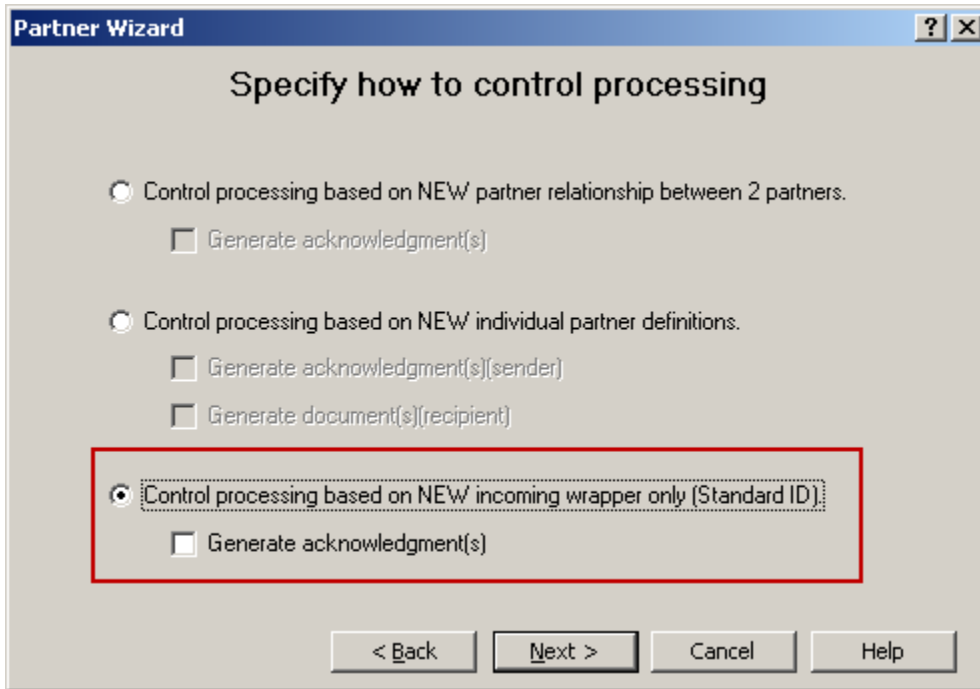
NOTE: If you already have a standard ID record that serves as a default for your incoming standard, you cannot use the Partner Wizard, because it only allows you to create new definitions. All you have to do is link the trade agreement you have already defined with your inbound Standard ID definition. To do this, you select the standard ID wrapper definition and add the trade agreement to the list on the **Trade Agreements** tab.

- 1 From the toolbar, select the Partner Wizard button . The Partner Wizard window appears.



- 2 Select **NEXT**.
The **Specify How to Control Processing** page appears.

- 3 Select the button **Control processing based on NEW incoming wrapper only (Standard ID)**. Do not select the check box beneath it, because we are only concerned here about receiving an acknowledgment.



- 4 Select **Next**.
The **Specify How to Identify the Incoming Standard** page appears.
- 5 For step 1, click the down arrow, and from the menu select **<Default>**.
We are going to use the default location for the inbound wrapper definition.

- 6 For step 2, click the down arrow, and from the menu select the inbound wrapper definition that matches the wrapper of the incoming acknowledgment.

Partner Wizard [?] [X]

Specify how to identify the incoming standard

1 Select an existing location, or enter a new location name.

Inbound (Source) Location: <Default>

2 Select a new or existing wrapper definition to be associated with this location

Inbound Wrapper Standard: EDIFACT, 2, UNBB

< Back Next > Cancel Help

- 7 Select **Next**.

The **Match Criteria** window appears.

Partner Wizard [?] [X]

Match Criteria

Add or modify matching criteria which will be used at runtime to identify the selected wrapper when data is received from the selected location.

Location: <Default> Standard: EDIFACT Version: 2 Wrapper ID: UNBB

Op	Offset	Segment	Field	Sub Field	Value
=	0	0	0	0	UNB
=	0	1	2	1	UNOB
=	0	1	2	2	2

Up Down New... Modify... Delete

< Back Next > Cancel Help

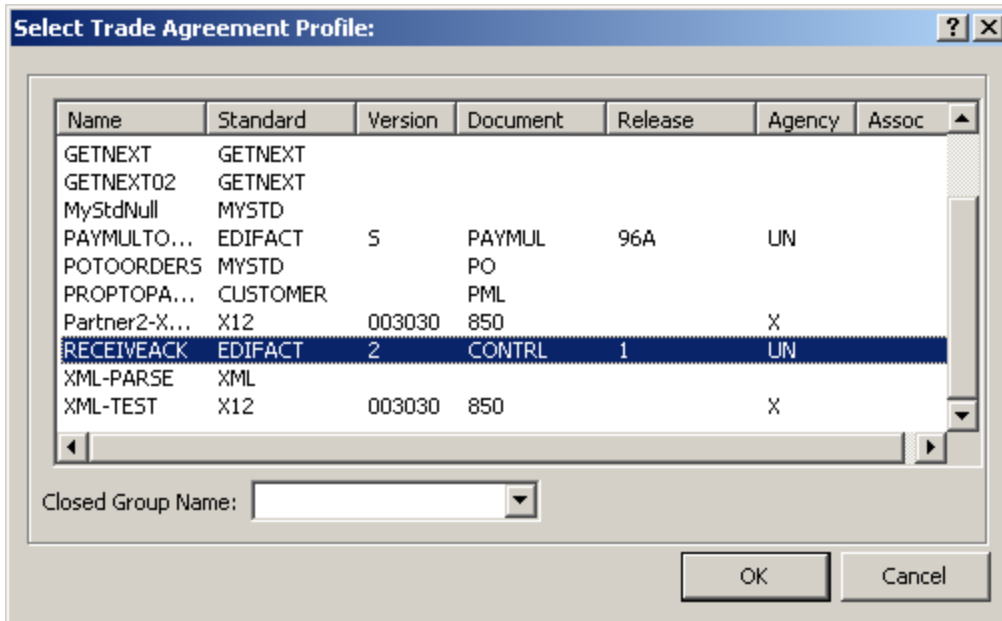
- 8 Modify the criteria, if you think it is necessary to allow the TRM to distinguish this wrapper from others in the <Default> location, and select **Next**.

The **Select Trade Agreement** dialog box appears.

- 9 Select the **Add** button.

The **Select Trade Agreement Profile** dialog box appears.

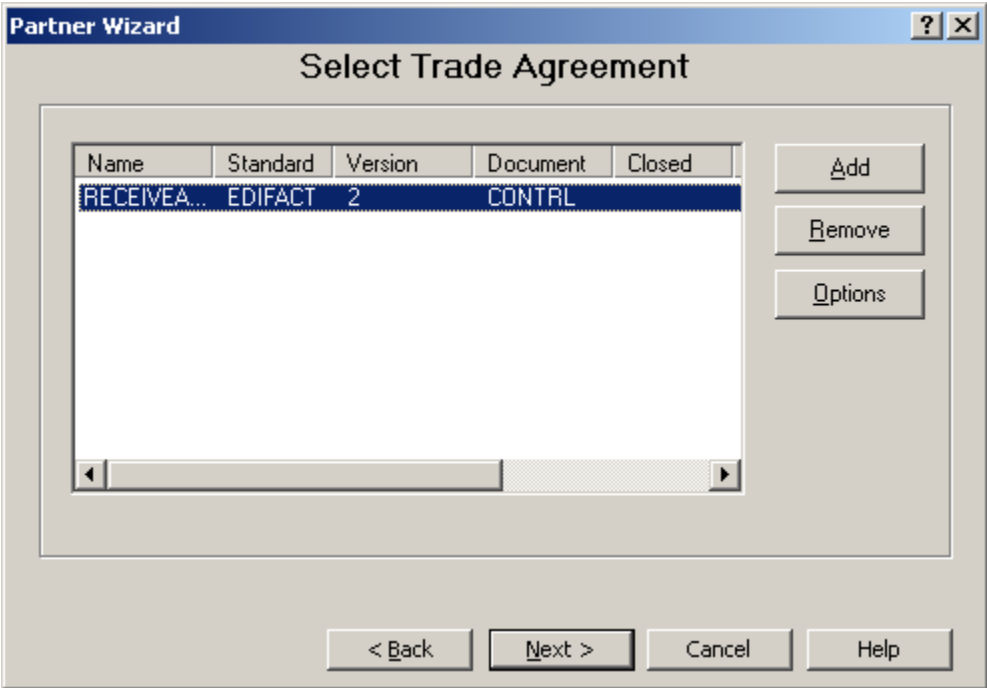
- 10 Select the appropriate trade agreement, and select **OK**.



The Standard ID Trade Agreement Options window appears, to allow you to create alternative IDs and location for the output. If you did not map the acknowledgment to create output, you will receive a warning message that no outputs are defined for this trade agreement.

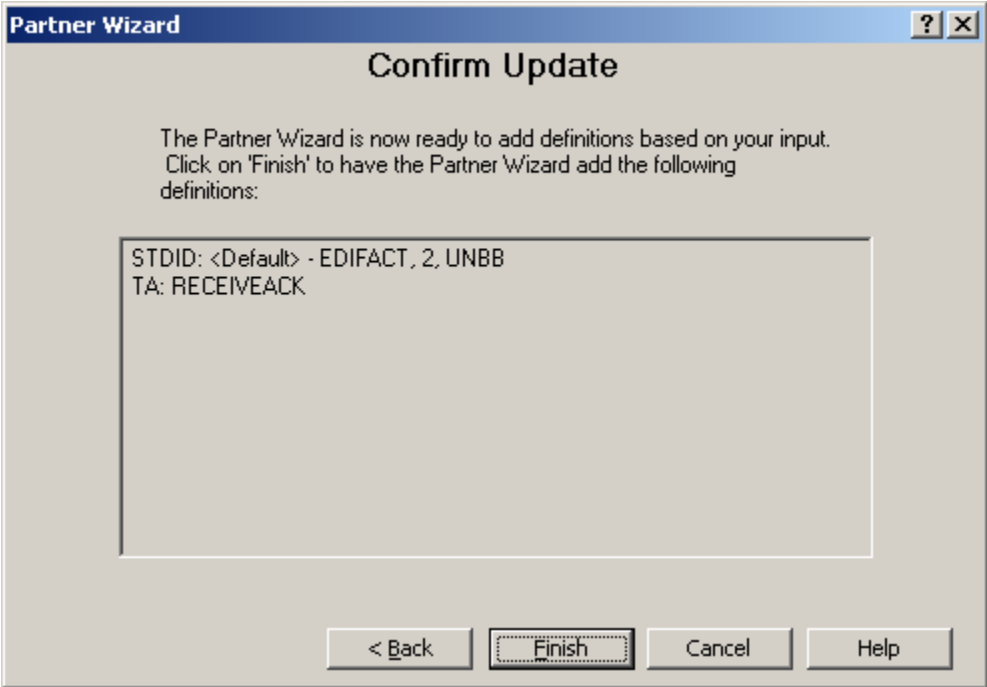
- 11 Select **OK** to exit.

The Select Trade Agreement window reappears.



12 Select **Next**.

The Confirm Update window appears, showing you the definitions that will be created based on what you have selected thus far.



13 Select **Finish**. When an information window appears, select **OK**.

This page intentionally blank

Validation and User Exits

Overview of Validate Routines and User Exits

MW Translator will check for compliance of inbound and outbound data with the definitions of the standards as follows:

Direction of Data	Validation Process	User Control
Inbound	Validate wrappers and/or contents	Trade Agreement Profile window, Options tab, Actions group, all action except Route File : <ul style="list-style-type: none"> ▪ Route File w/o Header and Validate Wrappers validate wrapper only ▪ Validate Contents and Translate validate wrappers and contents
	Validate element codes	<ul style="list-style-type: none"> ▪ Standard Version Properties window, General tab, Validate Element Codes box ▪ Wrapper Properties window, General tab, Validate Element Codes box ▪ Document Properties window, General tab, Validate Element Codes box
Outbound	Validate wrappers and contents	Trade Agreement Output Options (for Partner, Partner Relationship or Standard ID), Misc tab, Output Validation box

There are times that you may need to use Edibasic to access the data for further processing or to change the data in some way that you cannot do within MW Translator. Edibasic validation routines during the parse or mapping phases allow you to access the data, but not change it. User exits allow you great freedom to access and manipulate the data, but you must use C or C++ to write the user exit code. This section describes how to approach various tasks requiring Edibasic validation and user exits.

For more information about user exits, refer to the *MW Translator User Exits Programming Manual*.

Validating Data During Parsing

By default, MW Translator validates data as it parses the input document and wrapper. Input validation includes the following:

Validation Type	Description
Element type	Validate that the generated element matches the element type. Currently this involves character set validation. NOTE: It does not include validation of the element format, defined on the Standard Version window.
Character set	Validation checks that each character in the element is contained in the character set defined for the element type.
Width	Validate that the generated element satisfies the minimum and maximum width requirements of the definition.
Existence	Verify that mandatory elements, composites, segments and loops are present.
Repeat	Validate the number of generated elements, segments and loops are permitted by the definition.
Conditionals	Validate composite and segment conditions.
Segment sequence	Validate the sequence of segments against the appropriate configurations.
Control Reference	For enhanced control reference processing, validate the control reference received according to configurations in Control Reference Manager window.

When you want to do additional checking, you may add code to inbound document or wrapper definitions.

IMPORTANT: Validation uses the data values from the Data Element Store (DES), after the input data has been parsed, either by the TRM or by a pre-processing user exit.

You do this by adding Edibasic code to **Validate** or **ValidateAll** methods or by calling a user exit you have written to do so. There is an example of a user exit that validates a digital signature, which is part of the AUTACK example. For more information about this user exit, refer to the *MW Translator User Exits Programming Manual*.

Guidelines for Using Validate Methods during Parsing

Validate and **ValidateAll** methods may be attached to document and wrapper definitions in the Document, Wrapper, Segment, Composite Element and Element windows. Once defined, these user Edibasic methods become part of the parsing process of input objects. The TRM executes user-defined validate methods after parsing a given object, such as, document, wrapper, segment, or element.

You may invoke some actions whose results are different from the same action invoked in a map. For example, you can call the **Exception** function during parsing and mapping. If the function invokes reject during parsing, this action will result in a document level reject, unless the reject is performed in the wrapper. You cannot force a document-level reject during mapping, but you can set conditions in a map that terminate execution.

IMPORTANT: The result may be different if you place your Edibasic code in a **Validate** or **ValidateAll** method rather than in the map. Depending on where the validate routine is defined for the input definition (in the document, wrapper, segment, etc.) the parsing of the input document may not be complete and reference to the data element store may not succeed.

You can also override input data if the internal fields are defined on the segment or composite definitions, and have thus been stored in the Data Element Store, by using Edibasic **SetField()** statements in segment validate routines.

How to Use Validate Methods during Parsing

In general, where we refer to the Document window, the Wrapper window applies as well.

NOTE: To query values on an element or composite definition, you must use the **Element\$** function rather than the element name, because element and composite validation occurs during parsing before the segment is stored in the DES.

To Validate a Simple Element

The validation will repeat for every instance of the element.

- 1 Add the **Validate** method from the Element window or to the element detail from the Segment or Composite window.
- 2 Access the element value using the Edibasic function **Element\$**.

NOTE: If you define methods on both the Element window and on the Segment or Composite window, the method on the Segment or Composite window takes precedence over the method on the Element window. In no circumstances will two **Validate** methods be executed for a single element.

To Validate Several Values within a Composite Element

The validation will repeat for every instance of the composite.

- 1 Add the **Validate** method from the Composite window or to the element detail in the Segment window.
- 2 Access the various component values with the Edibasic function **Component\$**.

NOTE: If methods are defined on both the Composite window and the Segment window, then the method on the Segment window takes precedence over the method on the Composite window. In no circumstances will two **Validate** methods be executed for a single composite.

To Validate Several Element Values within a Segment

The validation will repeat for each instance of the segment.

- 1 Add the **Validate** method from the Segment window or to the segment detail in the Document window.
- 2 Access element values in the same way as you would in a map, for example, **N1{1,170}.1**.

NOTE: If methods are defined on both the Segment window and on the Document window, then the methods on the Document window takes precedence over the method on the Segment window. In no circumstances will two **Validate** methods be executed for a single segment.

To Validate Multiple Segments within a Single Iteration of a Loop

- 1 Add the **Validate** method to the loop level from the Document window.
- 2 Access the element values as you would in a map, for example, **N1{1,170}.1**. The validation will repeat for each instance of the loop.

To Validate All Instances of a Repeating Segment or Loop at Once

- Add the **ValidateAll** method to the loop or segment detail from the Document window. The **ValidateAll** method will execute once after all instances of the loop have been parsed.

To Validate Aspects of the Entire Document

- Add the **Validate** method to the Document window. The **Validate** method will execute after the entire document has been parsed.

Validating Data during Mapping

You may also validate data during mapping. This is where you can validate any input data, since you can be assured that the information has been placed in the data element store. Presumably, you will validate something other than inbound ID codes, because you can easily validate element ID codes during parsing using the global switches for an entire standard or for a specific document.

You can do this by adding Edibasic code to **Validate** or **ValidateAll** methods or by calling a user exit you have written to do so. For more information about how to use these methods, refer to the topic, *How to Use Validate Methods during Parsing* (on page 325).

There is an example user exit, **ValEle**, that validates element data in a document during mapping. You call this Edibasic user exit from an Edibasic **Validate** method within your map. You pass information between Edibasic methods and Edibasic user exits using global variables. For more information refer to the *MW Translator User Exits Programming Manual*.

Validating Data during Generation

To validate generated data before MW Translator completes processing, you select a check box on the **Misc** page of the Trade Agreement Options window for a partner, partner relationship or standard ID definition.

Partner Relationship Trade Agreement Options: EXAMPLE-X850 ? X

Output: 1,EXAMPLE,1,FPO 1 of 1

Sending Partner Recipient Partner **Misc**

Output Fields

Request Acknowledgment Test

Service Characters

Segment Terminator Component Delimiter Release Character

Tag Delimiter Element Delimiter Decimal Mark

Repeat Separator

Document Level Break Output Validation

Acknowledgment Expected

OK Cancel Apply

Output validation includes the following:

Validation Type	Description
Element type	Validate that the generated element matches the element type. Currently this involves character set validation. NOTE: It does not include validation of the element format, defined on the Standard Version window.
Character set	Validation checks that each character in the element is contained in the character set defined for the element type.
Width	Validate that the generated element satisfies the minimum and maximum width requirements of the definition.

Validation Type	Description
Existence	Verify that mandatory elements, composites, segments and loops are present.
Repeat	Validate the number of generated elements, segments and loops are permitted by the definition.
Conditionals	Validate composite and segment conditions.

User Exit Overview

The Workbench provides a high degree of flexibility in defining EDI document validation and/or translation through the use of Edibasic language to extend the definition-based validation and visual mapping capabilities of the Workbench. User exits further extend that flexibility by allowing the Translator Runtime Module (TRM) to exit into user code written in C or C++ during the validation or translation process. User exits are written in industry standard C or C++ using third party compilers. This provides the advantage of a full-featured, general-purpose computer language and allows access to resources on the target platform, such as files or databases.

There are seven types of user exits:

- An **audit** user exit allows custom audit logs to be generated or allows event monitoring.
- **Edibasic** user exits further extend validation and mapping.
- **Security** user exits provide the ability to review and calculate hash values for input and output data.
- **Pre-processing** user exits provide inline pre-processing of the input stream.
- **Post-processing** user exits provide inline post-processing of the output stream.
- **Control reference validation** user exits allow customized validation of control references.
- **Control reference generation** user exits allow customized generation of control references.

You write user exits as C functions and compile them using a third party compiler. A register function is also written to register the user exits with the TRM. The user exit(s) and the register function are then statically or dynamically linked to the TRM, depending on the target platform. For more information about user exits, refer to the *MW Translator User Exits Programming Manual*.

User Exit Examples

To help you understand the process, we have provided examples of user exits with the Workbench. There are examples for each type of user exit.

The subdirectories within the **EXAMPLES** subdirectory that pertain to user exits are listed in the following table. Note that some of the subdirectories contain examples that you can load and run, but do not use user exits, such as **X850Test**, **SWIFT**, and **XML**.

Subdirectory	Type	Description
AddTag	pre-processing	Contains an example of a pre-processing user exit that adds a tag to a segment before the map processes the data.
Audit	audit	Contains an example of an audit user exit that logs all audit events to a file.
AUTACK	security, Edibasic	Contains an example of a security user exit that calculates hash values for input or output data. For validation of incoming data, it is coupled with an Edibasic user exit called from a Validate Method in the input document definition to validate a digital signature. For generation of outgoing data, it is coupled with an Edibasic user exit called from a map to generate a digital signature.
Projects	NA	For your convenience, this folder is a place holder for your user exit projects.
RemSpc	post-processing	Contains an example of a post-processing user exit that removes trailing spaces after the map processes the data.
Source	NA	Contains source code for all user exits
ValEle	Edibasic	Contains an example of an Edibasic user exit that uses global variables to pass information between a map and an external program.

IMPORTANT: When you write a control reference user exit, you must also configure a setting in the MW Translator Operator program. You must select the **Control Reference Processing** option on the **Server Options** tab of the Options for MessageWay window.

Configuring a security user exit provides an opportunity to review many features and procedures that help you understand all user exits. To follow a complete configuration process, refer to the information about security user exits.

Security User Exit (AUTACK) Overview

The EDIFACT AUTACK message has two different functions: one for authentication and non-repudiation of origin, and one for acknowledgment and non-repudiation of receipt. The implementation of AUTACK supports only the authentication function.

AUTACK is a specific type of summary document. Namely, it is used here to facilitate the security services required to authenticate other non-AUTACK documents within the same interchange or provide non-repudiation of origin.

The TRM will process any documents within an interchange defined within the partnership configurations to be secured with an AUTACK message. For incoming documents that require security validation using AUTACK, the TRM will perform the validation, and may return a CONTRL document secured with an AUTACK message. For outgoing documents that require security, the TRM will generate messages within an interchange secured with an AUTACK message, and will process any responding CONTRL documents that may also be secured with an AUTACK message.

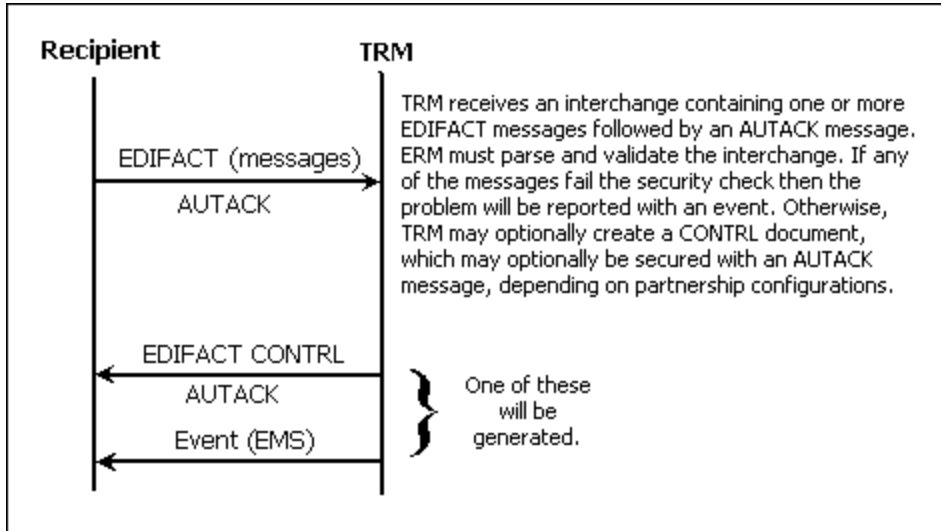
The software provides a framework for AUTACK validation and generation. The user must provide the actual implementation through user exits to perform the actual security algorithms supported by appropriate configurations. Example user exits and configurations based on simplistic (unsecured) algorithms are provided to be used as an implementation guideline.

Processing AUTACK Messages

The TRM processes AUTACK messages based on configurations that you have provided for various entities and on the user exits that you write to support the security processing requirements. We will view how the TRM processes such information from 2 perspectives: when it receives secured messages and when it sends secured messages. This is because your tasks are typically associated with an entire information flow. We describe the 2 compound exchanges of information separately, because each requires different configurations, although they both use the same user exits you provide, and some of the same configurations. Typically, these would represent 2 separate EDI processing systems, but we have defined them together here for expediency. Later we will discuss these exchanges using the security user exit examples that are provided with the workbench.

Receiving Secured Messages and Sending Secured Acknowledgments

The TRM processes the incoming message(s) within an interchange that are secured with an AUTACK message. Depending on partnership configurations, the TRM may generate a CONTRL document that may also be secured with an AUTACK message. The following diagram gives you an overview of the process.



While parsing the inbound EDIFACT messages, the TRM will call a user exit to process the inbound message data. The user exit will scan for messages (other than AUTACK) and either directly or indirectly (via a third party security package) computes a hash result for those messages. The user exit will compute the hash result starting with the UNH of the first message and ending with the segment terminator of the UNT of the last message. The TRM saves the hash results for later processing.

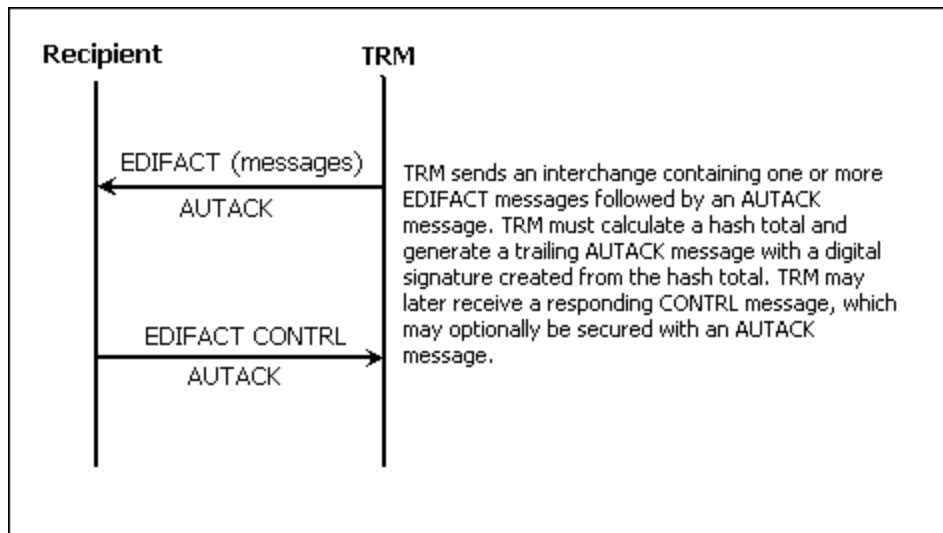
While parsing the inbound AUTACK message, a validation user exit will be called to verify security for the interchange. This user exit will verify the digital signature contained in the AUTACK using the sender's public key and the saved hash results. A security exception will be generated if an error is encountered. In the event of a security error, an EMS event will be generated and the entire interchange will be rejected.

If the AUTACK message is missing from the interchange, and it is defined to be there, then a security error will be generated. Configuration options in the Trade Agreement Profile for the subject message determine if an AUTACK message should be present.

If no security errors are encountered then the TRM, if so configured, will generate a CONTRL message. The CONTRL message may be secured with an AUTACK message. Configuration options in the CONTRL Acknowledgment profile determine if the TRM generates an AUTACK message. Like current acknowledgments, the AUTACK will be generated with a map. While generating the CONTRL message, a user exit will be called to calculate a hash value. The AUTACK map will call a user exit to create a digital signature from the previously calculated hash value. The digital signature will be encoded in ASCII and generated in the USY segment.

Sending Secured Messages and Receiving Secured Acknowledgments

The TRM generates the outgoing message(s) within an interchange and secures them with an AUTACK message. The TRM validates any responding CONTRL document that may also be secured with an AUTACK message. The following diagram gives you a quick view of the process.



While generating the outbound EDIFACT messages, the TRM will call a user exit to process the outbound message data. The user exit will either directly or indirectly (via a third party security package) compute a hash result of messages other than AUTACK. The user exit will scan the outbound data and calculate a hash result starting with UNH segment of the first message and ending with the segment terminator of UNT segment of the last message. The hash result will be saved for later processing.

Before generating the outbound wrapper trailer, the TRM will generate an AUTACK message based on options in the subject message Trade Agreement Profile. The AUTACK message will be generated with a map in a similar fashion to an acknowledgment. The map will call a user exit to create a digital signature from the previously calculated hash value. The digital signature will be encoded in ASCII and generated in the USY segment.

While parsing the inbound CONTRL message, the TRM will call a user exit to process the inbound message data. The user exit will scan for messages (other than AUTACK) and either directly or indirectly (via a third party security package) computes a hash result. The user exit will compute the hash result for the CONTRL message starting with the UNH segment and ending with the segment terminator of the UNT segment. The hash results will be saved for later processing.

While parsing the inbound AUTACK message, a validation user exit will be called to verify security for the inbound CONTRL. This user exit will verify the RSA digital signature contained in the AUTACK using the sender's public key and the saved hash results from the subject message (CONTRL). A security exception will be generated if an error is encountered. In the event of a security error, an EMS event will be generated and the entire interchange will be rejected.

If the AUTACK is missing from the interchange, then a security error will be generated.

Checklist of Prerequisite Configurations

This discussion assumes that you are adding security processing to existing definitions. Therefore, you would already have configured and tested certain configurations. A separate **README** file describes how to load the definitions that support the examples. These definitions support both exchange processes discussed here. Typically, these configurations would exist as needed on separate EDI systems to support a specific viewpoint: that of the sender or that of the recipient. We have put them together here on one EDI system, represented by your Workbench, for expediency. Once loaded, the following configurations should be in your workbench:

Definition Type	Definition Names	Description
Wrappers	BANKWRAP CUSTWRAP UNBA	BANK, 1, BANKWRAP CUSTOMER, 1, CUSTWRAP EDIFACT, 2, UNBA

Definition Type	Definition Names	Description
Documents	APPPML AUTACK CONTRL PAYMUL PROPPML	BANK, 1, APPPML EDIFACT, 4, AUTACK EDIFACT, 2, CONTRL EDIFACT, S96.A, PAYMUL CUSTOMER, 1, PROPPML
Maps	AUTACK BANKWRAP CONTRL2 PAYMULTOPROP PROPTOPAYMUL RCNCL UNBA2	Map to generate AUTACK. Proprietary wrapper map. Map to generate EDIFACT CONTRL. This contains new logic to support security errors. EDIFACT PAYMUL to BANK PML map. CUSTOMER PML to EDIFACT PAYMUL map. Map to copy CONTRL to a document for external reconciliation. EDIFACT wrapper map
Trade Agreement Profiles	AUTACK PAYMULTOAPP PROPTOPAYMUL RECEIVEACK	Trade agreement for AUTACK documents. Trade agreement to map the EDIFACT PAYMUL to the proprietary BANK PML. Trade agreement to map the proprietary CUSTOMER PML to EDIFACT PAYMUL. Trade agreement for receipt of CONTRL documents.
Acknowledgment Profiles	SENDACK	Returns CONTRL document to sender.
Standard Identification Locations and wrappers	CUSTOMER EDIFACT	CUSTOMER, 1, CUSTWRAP EDIFACT, 2, UNBA

Definition Type	Definition Names	Description
Partner Profiles	CUSTOMER	Sends EDIFACT PAYMUL, secured with AUTACK to BANK. Receives EDIFACT CONTRL, secured with an AUTACK.
	BANK	Receives EDIFACT PAYMUL, secured with AUTACK. Generates CONTRL, secured with AUTACK.
	RECONCILE-APP	Partner location where CONTRL messages are routed for reconciliation processing.

Checklist of User Exit Functions

The user must create user exits to implement the AUTACK security requirements, which are supported by a TRM framework. The following is a list of registered user exits and their functions within the user exit file DIGSIGN32.DLL.

User Exit Name	User Exit Type	Function
BCEHash	Security	calculates the hash value associated with the document for which it is defined in the Trade Agreement Profile or Acknowledgment Profile
genBCESign	Edibasic	generates digital signature for outgoing AUTACK
valBCESign	Edibasic	validates digital signature for incoming AUTACK

IMPORTANT: It is a good idea to combine related user exits in a single file, because each environment is limited to 10 user exit DLL files during processing. However, there may be only one security user exit per input stream or output stream.

Configuring Definitions to Receive Secured Messages

The following discussion uses the examples provided with the new functionality: an exchange between a customer and a bank from the viewpoint of the recipient, BANK. It steps you through the configuration process required to create the example configurations.

Assume that the primary tasks of this inbound processing are as follows:

- 1 Parse all incoming messages within an interchange.
- 2 For each document parsed in step 1 that is not an AUTACK message and for which a security user exit is defined, call the security user exit to calculate a hash value, passing data one segment at a time.

- 3 Parse the AUTACK message and call the validation user exit identified in the document definition to verify the digital signature using the sender's public key and the hash value calculated with the security user exit in step 2.
- 4 Respond with a CONTRL document that is secured with an AUTACK message (optional).

Configuration Tasks to Receive Secured Messages and Return Secured Acknowledgment

These are the basic steps you would follow to modify your existing configurations. The steps listed here have already been done, and the results are included in the configurations you loaded for the example. Step 1 is a generic step that supports both inbound and outbound processes. Steps in the topic, *To Configure Inbound Processing to Parse and Authenticate PAYMUL Secured with AUTACK* (on page 338), support processing of the inbound EDIFACT document, PAYMUL. Steps in the topic, *To Configure Outbound Processing to Generate CONTRL Secured with AUTACK* (on page 338), support generation of the EDIFACT acknowledgment, CONTRL. We will use the information from the security user exit example to provide a more detailed explanation.

Generic Step

- 1 Write a program containing 3 functions to be registered as a single user exit: One security user exit to calculate running hash values and two Edibasic user exits, one to verify a digital signature and another to generate a digital signature.

To Configure Inbound Processing to Parse and Authenticate PAYMUL Secured with AUTACK

- 1 Specify the security document that should accompany the PAYMUL and the security user exit that will calculate the hash value, which is based on the segments in the PAYMUL.
- 2 Add a **Validate** method to the AUTACK definitions that calls the Edibasic user exit (**valBCESign**) to verify the digital signature.

To Configure Outbound Processing to Generate CONTRL Secured with AUTACK

- 1 In the trade agreement profile, specify the summary document (AUTACK) to accompany the acknowledgment.
- 2 Specify the security user exit to calculate the hash total for the CONTRL acknowledgment.
- 3 Add an Edibasic user exit (**genBCESign**) for generating a digital signature to the AUTACK map.

Writing the User Exits

You will need to write three user exits using C or C++: a security user exit to calculate the hash totals, an Edibasic user exit called from a **Validate Method** to validate the digital signature, and an Edibasic user exit called from the AUTACK map to generate the digital signature. For additional information about user exits in general and Edibasic and security user exits in particular, refer to the *MW Translator User Exits Programming Manual*.

Please note the following requirements when creating your user exits:

- You can have only one security user exit per input stream or output stream.
- When writing the Edibasic user exit to validate the digital signature in the security document, you must explicitly call the function TRMSecurityStatus and pass it an action parameter. If you do not do this, the TRM generates a security exception.
- You register all user exits by calling the appropriate TRM register function, as shown in the following sample code.

```
ERMRegisterSecExit ("BCEHash", BCESecurityHash) ;  
ERMRegisterUserExit ("genBCESign", BCEGenerateSign) ;  
ERMRegisterUserExit ("valBCESign", BCEValidateSign) ;
```

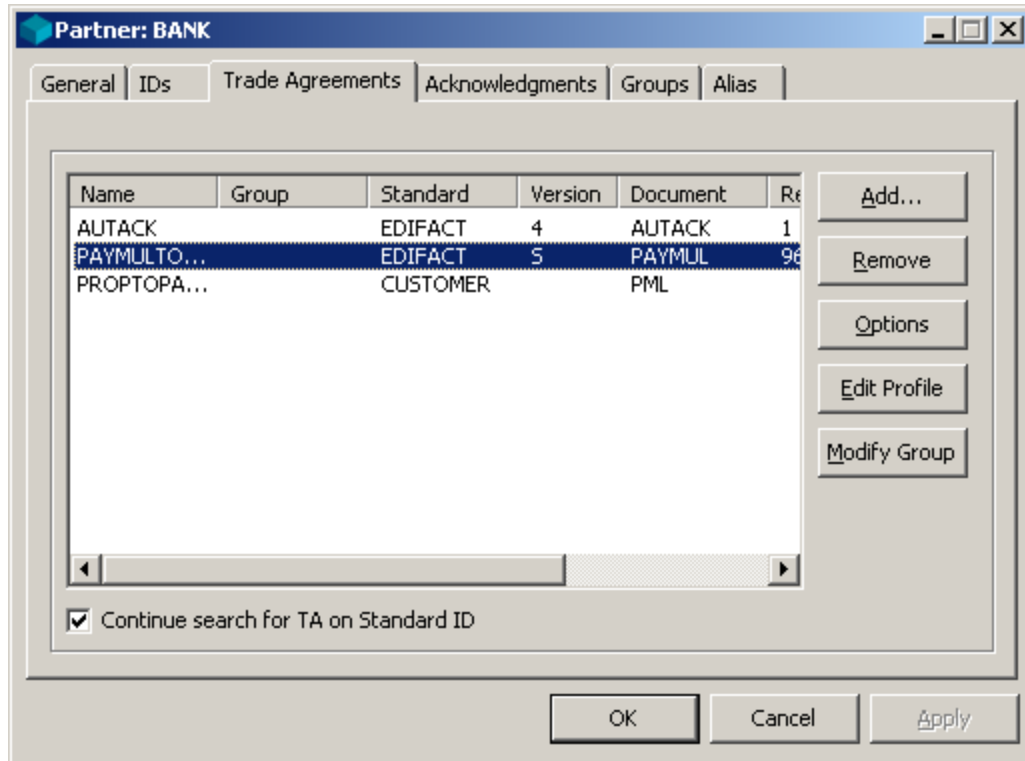
The first parameter of each function call will be the user exit name you specify in the security user exit fields on the trade agreement profile and the acknowledgment profile, or in the **Validate Method** of the input document definition, or in the map for the security document.

Refer to the documentation in the sample program, DIGSIGN.CPP, for a more detailed explanation of coding issues.

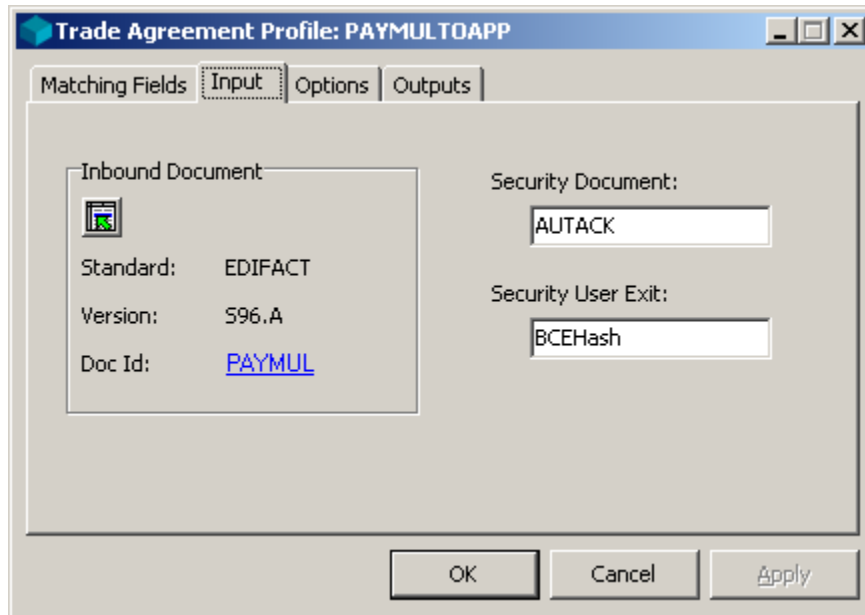
Specifying the Security Document and Security User Exit for Incoming Document(s)

For the recipient partner, configure the trade agreement profile for the incoming document(s) to identify the security user exit and the security document, AUTACK.

The following screen identifies the appropriate trade agreement profiles, PAYMULTOAPP and AUTACK, for the recipient partner, BANK.



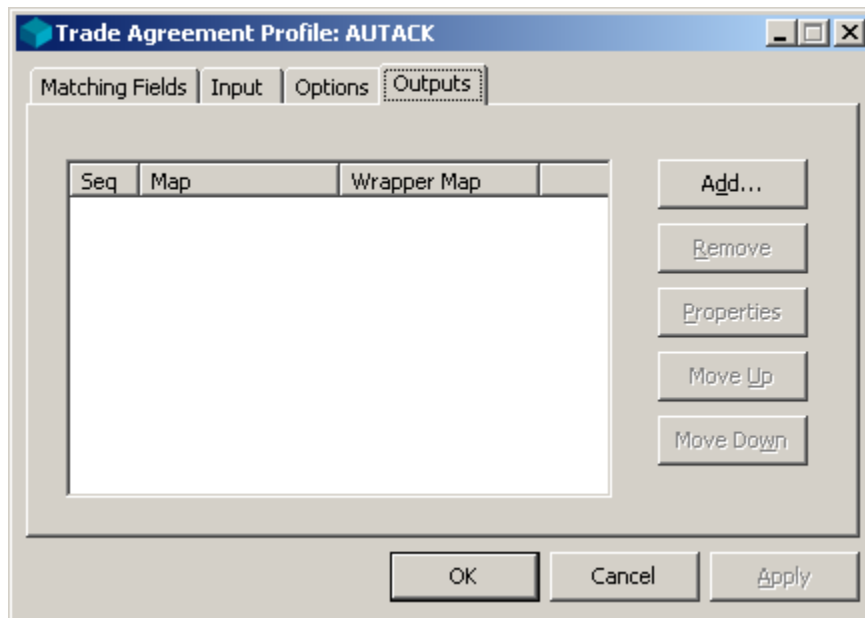
The following screen shows you the trade agreement profile for the incoming document, PAYMUL, and identifies the security document, AUTACK, that should accompany the PAYMUL. It also identifies the security user exit, **BCEHash**, that will be called to calculate the hash value.



The following screens show you the trade agreement, AUTACK, that is received with the PAYMUL. The TRM must parse the contents to be able to execute the Edibasic user exit that validates the digital signature. To accomplish this, you could select either **Validate Contents** or **Translate** from the **Options** tab. However, since you do not need any output from the AUTACK, you can select **Translate** and not put anything on the list on the **Outputs** tab. When you select **Validate Contents**, the TRM generates an exact copy of the input for output routing.

The screenshot shows the 'Trade Agreement Profile: AUTACK' dialog box with the 'Input' tab selected. The 'Inbound Document' section contains a file icon, 'Standard: EDIFACT', 'Version: 4', and 'Doc Id: AUTACK'. The 'Security Document' and 'Security User Exit' sections each have an empty text input field. At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

The screenshot shows the 'Trade Agreement Profile: AUTACK' dialog box with the 'Options' tab selected. The 'Action' section has radio buttons for 'Route File', 'Route File w/o Header', 'Validate Wrappers', 'Validate Contents', and 'Translate' (which is selected). The 'Error Action' section has radio buttons for 'Accept' and 'Reject' (which is selected). At the bottom are 'OK', 'Cancel', and 'Apply' buttons.



Specifying the User Exit to Validate the Digital Signature

For the AUTACK document definition, write a validation method to call the Edibasic user exit that validates the digital signature. The user exit validates the digital signature, returning accept or reject status to the TRM. The following screens show you the code, but you can also use Print Preview to see the Edibasic code associated with the **Validate Method**. Note that the global variable, **digSign**, is used to transfer the digital signature from Edibasic to the user exit code. The global variable is declared on the USY segment.

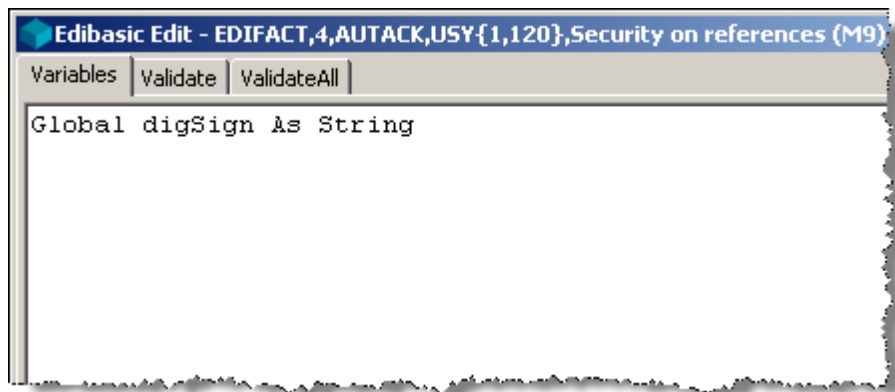
NOTE: A check mark in the column indicates that some Edibasic code is associated with the entity. You can right-click the marked selections in the pop-up menu to review the code.

Document: EDIFACT, 4, AUTACK

Description: Authorization and Acknowledgment

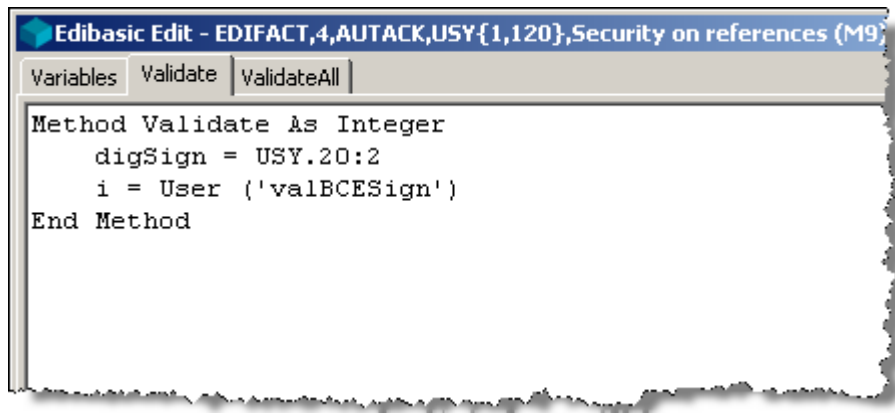
L/S	Level	Area	Seq	Tag	Description	Rqmt	Max C
S	0	1	10	UNH	MESSAGE HEADER	M	
L	0	1	20	SG1	Loop	M	
S	1	1	30	USH	SECURITY HEADER	M	
S	1	1	40	USA	Security algorithm	C	
L	1	1	50	SG2	Loop	C	
S	2	1	60	USC	Certificate to convey public key and the credentials of its own	M	
S	2	1	70	USA	Security algorithm	C	
S	2	1	80	USR	Security Result	C	
S	0	1	90	USB	Secured data identification (M1)	M	
L	0	1	100	SG3	Loop	M	9
S	1	1	110	USX	Security references	M	
S	1	1	120	USY	Security on references (M9)	M	
L	0	1	130	SG4	Loop	M	
S	1	1	140	UST	Security trailer (M1)	M	
S	1	1	150	USR	Security Result	C	
S	0	1	160	UNT	MESSAGE TRAILER	M	

- Variables
- Validate
- ValidateAll
- Ack Wrapper Level
- Ack Document Level
- Insert Row
- Delete Row



The screenshot shows a window titled "Edibasic Edit - EDIFACT,4,AUTACK,USY{1,120},Security on references (M9)". The window has a menu bar with "Variables", "Validate", and "ValidateAll". The main text area contains the following code:

```
Global digSign As String
```



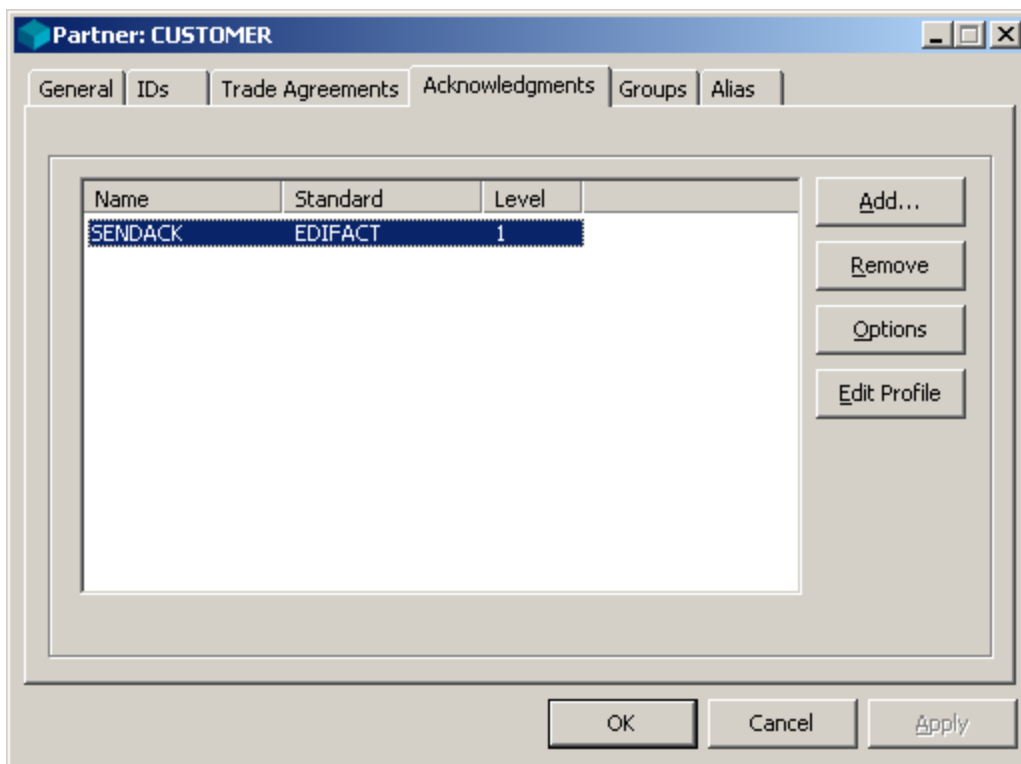
The screenshot shows a window titled "Edibasic Edit - EDIFACT,4,AUTACK,USY{1,120},Security on references (M9)". The window has a menu bar with "Variables", "Validate", and "ValidateAll". The main text area contains the following code:

```
Method Validate As Integer  
    digSign = USY.20:2  
    i = User ('valBCESign')  
End Method
```

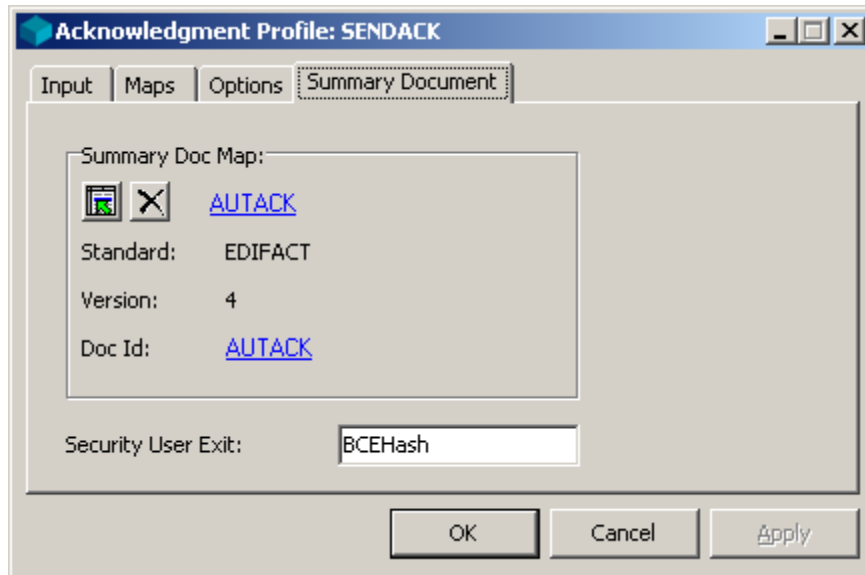
Specifying the User Exit to Calculate the Hash Value for the Summary Document

You should already have the partner profile for the sending partner configured to return an acknowledgment. The acknowledgment profile should identify the security user exit that will create the hash value based on the acknowledgment segments. You then map the value to the appropriate location in the accompanying security document, the AUTACK.

The following screen shows you the partner profile for the sending partner, CUSTOMER, which points to the appropriate acknowledgment profile, SENDACK.




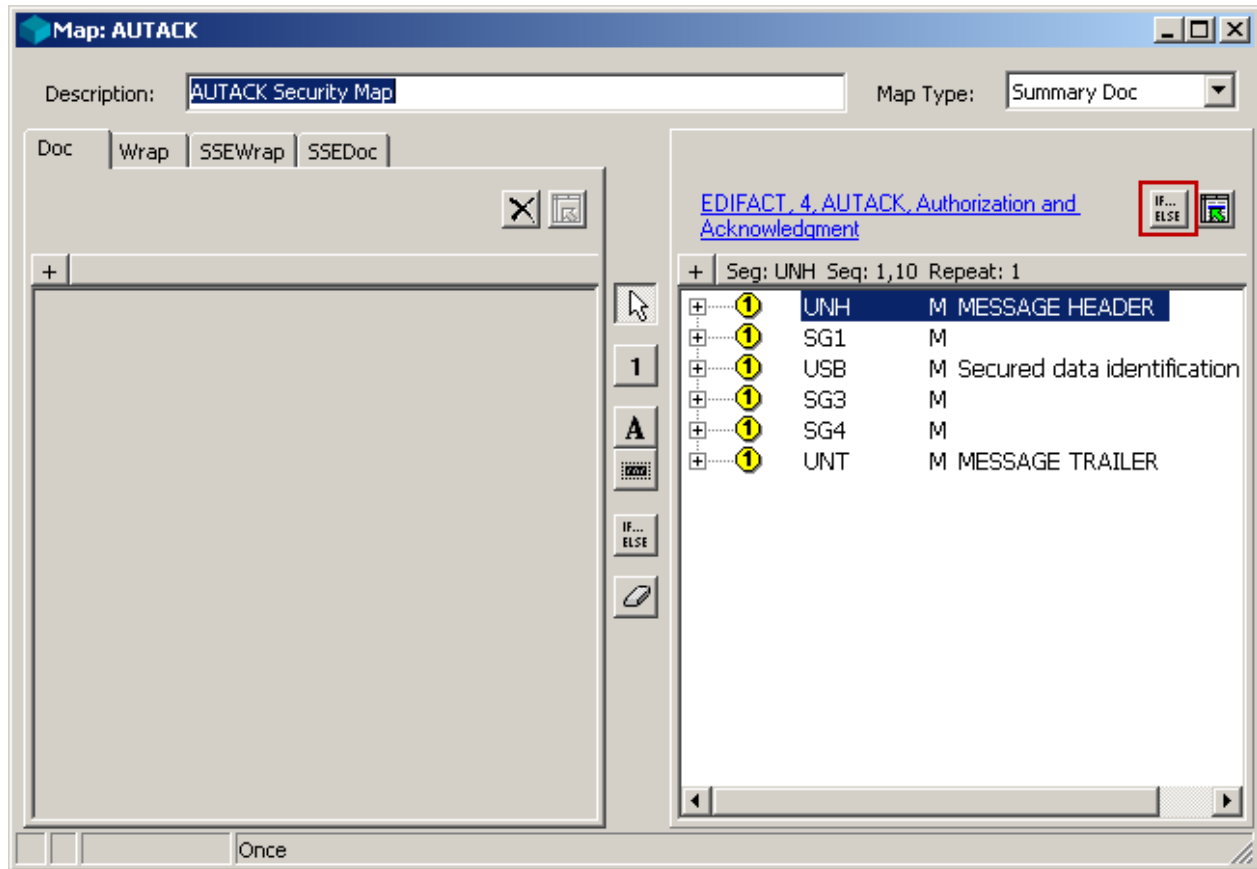
The following screen shows you the appropriate acknowledgment profile, SENDACK, associated with the sender of the PAYMUL document, CUSTOMER. It identifies the map that will create the summary document, the AUTACK. It also identifies the security user exit that will be called to calculate the hash value based on the segments in the acknowledgment.



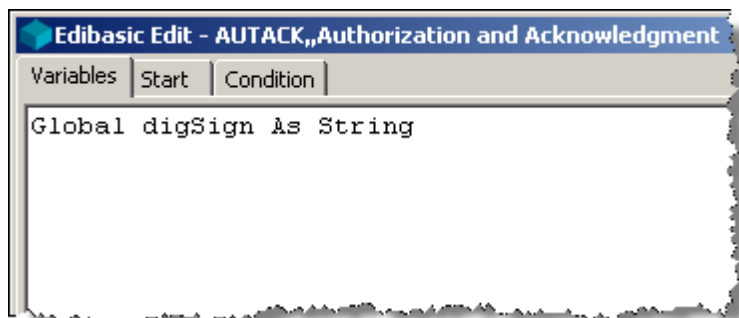
Specifying the User Exit to Generate the Digital Signature

For the AUTACK map, you write an Edibasic method to call the Edibasic user exit that uses a public key and the hash value to generate a digital signature. Place the signature in the AUTACK message that will accompany the acknowledgment. Note that the user exit saves the generated digital signature in that global variable, **digSign**.

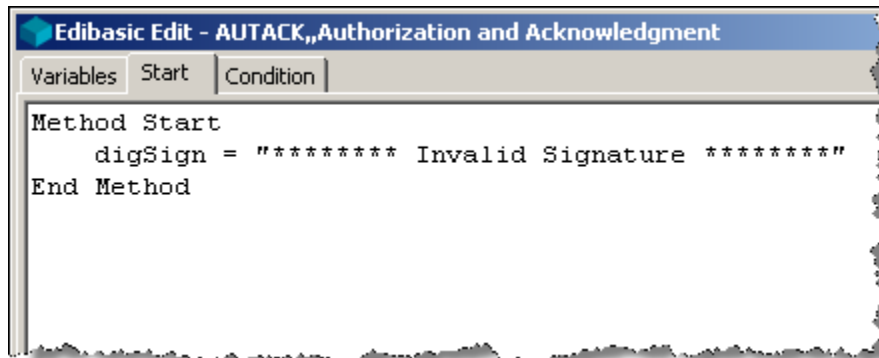
The global variable is defined at the document level. You can review this code by selecting the **Document Methods and Variables** button  next to the destination document header.



The variable is declared on the **Variables** tab.



Note that the value is initialized on the **Start** tab.

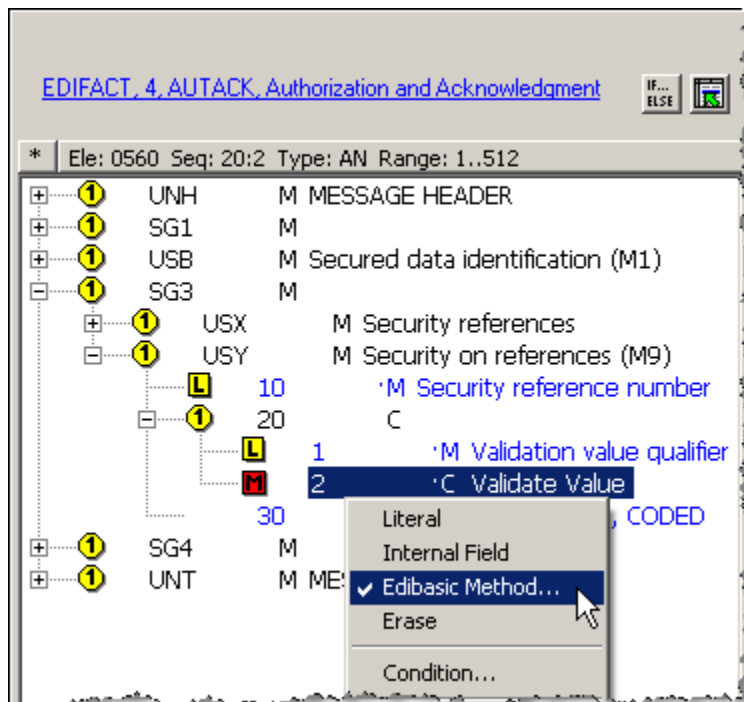


```
Edibasic Edit - AUTACK,,Authorization and Acknowledgment
Variables Start Condition
Method Start
  digSign = "***** Invalid Signature *****"
End Method
```

The following screens show you the call to the user exit. You can also select the **Print Preview** button



from the toolbar to view the code.



EDIFACT, 4, AUTACK, Authorization and Acknowledgment

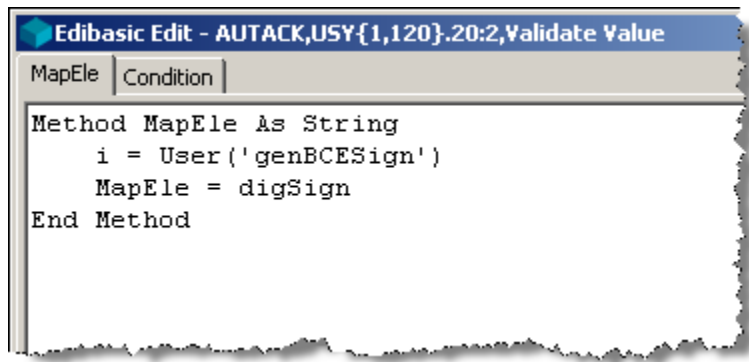
IF... ELSE

* Ele: 0560 Seq: 20:2 Type: AN Range: 1..512

UNH	M	MESSAGE HEADER
SG1	M	
USB	M	Secured data identification (M1)
SG3	M	
USX	M	Security references
USY	M	Security on references (M9)
10	M	Security reference number
20	C	
1	M	Validation value qualifier
2	C	Validate Value
30		, CODED
SG4	M	
UNT	M	ME

Context menu options:

- Literal
- Internal Field
- Edibasic Method...
- Erase
- Condition...



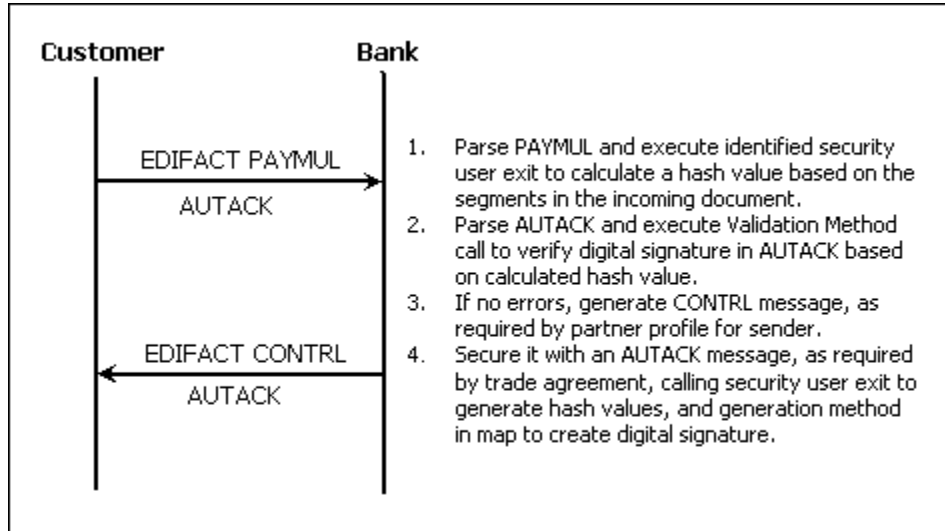
```
Edibasic Edit - AUTACK,USY{1,120}.20:2,Validate Value
MapEle | Condition |
Method MapEle As String
  i = User('genBCESign')
  MapEle = digSign
End Method
```

Files for AUTACK Example

Three sample input files have been included that reflect this type of exchange as explained in the following table. Refer to the **README** file included with the example for more information.

Input file	Description
inpaymul.txt	Processing includes normal incoming PAYMUL secured with AUTACK; generates an outgoing CONTRL secured with an AUTACK.
inerr.txt	Processing includes incoming PAYMUL secured with an AUTACK message that has a invalid digital signature, generates an outgoing CONTRL secured with an AUTACK
innoatck.txt	Processing includes incoming PAYMUL that is missing the accompanying AUTACK; generates an outgoing CONTRL secured with an AUTACK.

The following diagram reflects the prototype of the example included in the AUTACK subdirectory. It shows the document exchange between a customer and the bank, from the viewpoint of the bank.



The following definitions are used to support the incoming EDIFACT PAYMUL message secured with an AUTACK message:

Definition Type	Definition Names	Description
Wrappers	BANKWRAP UNBA	Used to generate proprietary bank wrapper. Used to parse EDIFACT wrapper.
Documents	APPPML AUTACK PAYMUL	Used to generate proprietary APPPML document Used to parse EDIFACT AUTACK message and call user exit to verify digital signature. Used to parse EDIFACT PAYMUL message.
Maps	BANKWRAP PAYMULTOPROP UNBA2	Used to generate proprietary wrapper. Used to generate proprietary PML document. Used to generate EDIFACT wrapper
Trade Agreement Profiles	AUTACK PAYMULTOAPP	Trade agreement for incoming AUTACK documents. Trade agreement to generate the proprietary BANK PML from the EDIFACT PAYMUL.
Standard Identification Locations and wrappers	EDIFACT	EDIFACT,2,UNBA

Definition Type	Definition Names	Description
Partner Profiles	CUSTOMER	Receives EDIFACT CONTRL, secured with an AUTACK.
	BANK	Receives EDIFACT PAYMUL, secured with AUTACK.

The CONTRL message is generated as part of the processing of the PAYMUL message. It uses some of the configurations listed for the PAYMUL processing, but for different purposes, and some additional configurations, which are all listed here:

Definition Type	Definition Names	Description
Wrappers	UNBA	Used to generate EDIFACT CONTRL wrapper.
Documents	AUTACK	Used to generate EDIFACT AUTACK message.
	CONTRL	Used to generate EDIFACT CONTRL message.
Maps	UNBA2	Used to generate EDIFACT wrapper.
	CONTRL2	Used to generate EDIFACT CONTRL. This contains new logic to support security errors.
	AUTACK:	Used to generate EDIFACT AUTACK.
Acknowledgment Profiles	SENDACK	Returns CONTRL document to sender (identified on CUSTOMER partner profile).
Partner Profiles	CUSTOMER	Specifies acknowledgment profile to receive EDIFACT CONTRL, secured with an AUTACK.
	BANK	Sends EDIFACT CONTRL, secured with AUTACK.

Configuring Definitions to Send Secured Messages

The following discussion uses the examples provided with the new functionality: an exchange between a customer and a bank from the viewpoint of the sender, CUSTOMER. Other implementations may vary. The primary tasks of this outbound processing are as follows:

- 1 Parse incoming proprietary documents.
- 2 For each document for which a security user exit is defined, call the security user exit to calculate a hash value, passing data one segment at a time.
- 3 After having generated all documents within an interchange, generate the security document, AUTACK.

- 4 Parse and authenticate responding, incoming CONTRL document that is secured with an AUTACK message.

Configuration Tasks to Send Secured Messages and Receive Secured Acknowledgment

These are the basic steps you would follow to modify your existing configurations. The steps listed here have already been done, and the results are included in the configurations you loaded for the example:

- Step 1 is a generic step used by both outbound and inbound processes. Steps in the topic, *To Configure Outbound Processing to Generate PAYMUL Secured with AUTACK* (on page 353), support processing of the outbound EDIFACT document, PAYMUL.
- Steps in the topic, *To Configure Inbound Processing to Parse and Authenticate CONTRL Secured with AUTACK* (on page 353), support validation of the responding EDIFACT acknowledgment, CONTRL.
- The last step in the topic, *To Route the CONTRL Document to Alternate Location for Application Processing* (on page 354), is optional, and routes the CONTRL document.

We will use the information from the security user exit example to provide a more detailed explanation.

Generic Step

- 1 Write a program containing 3 functions to be registered as a single user exit: One security user exit to calculate running hash values and two Edibasic user exits, one to verify a digital signature and another to generate a digital signature.

To Configure Outbound Processing to Generate PAYMUL Secured with AUTACK

- 1 Specify the security user exit that will calculate the hash value (**BCEHash**), which is based on the segments in the proprietary payment document, and specify the summary document (map) (**AUTACK**) to accompany the outbound EDIFACT PAYMUL document.
- 2 Add an Edibasic user exit (**genBCESign**) to the AUTACK map to generate a digital signature.

To Configure Inbound Processing to Parse and Authenticate CONTRL Secured with AUTACK

- 1 Specify the security document and security user exit for incoming acknowledgment (**CONTRL**).
- 2 Specify the Edibasic user exit (**valBCESign**) in the AUTACK document definition to authenticate the digital signature.

To Route the CONTRL Document to Alternate Location for Application Processing

- 1 Use aliasing to route the control document to a location other than the location specified by the sending partner profile.

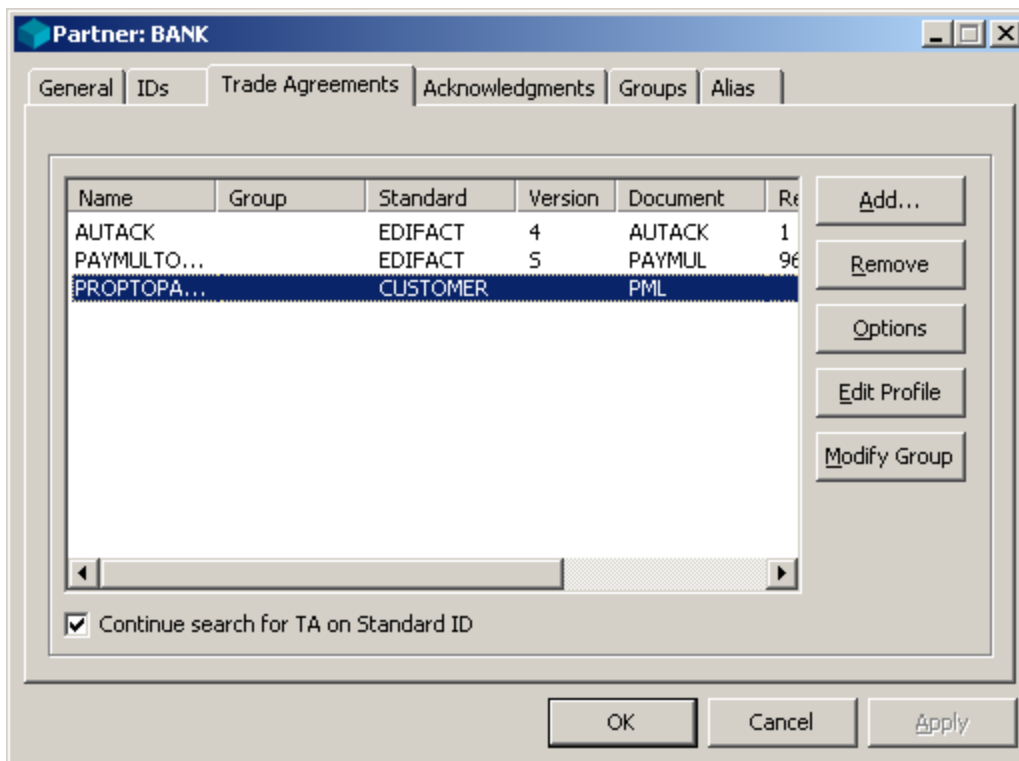
Writing the User Exits

This is the same step described previously, *Writing the User Exits* (on page 339).

Specifying the Security User Exit and the Summary Document for the Outgoing Document

For the recipient partner, BANK, configure the Trade Agreement Profile for the outgoing document(s) to identify the security user exit and the summary document, AUTACK.

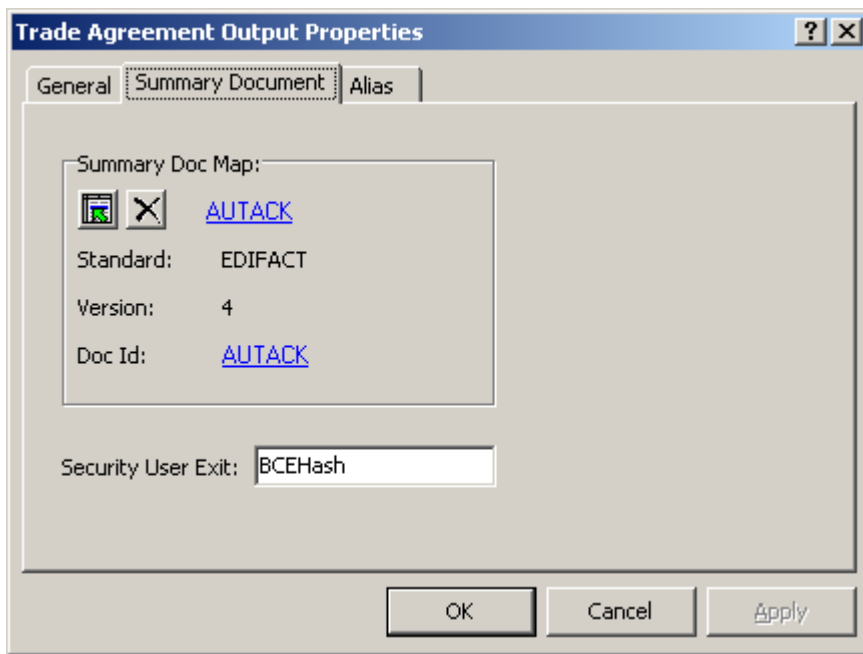
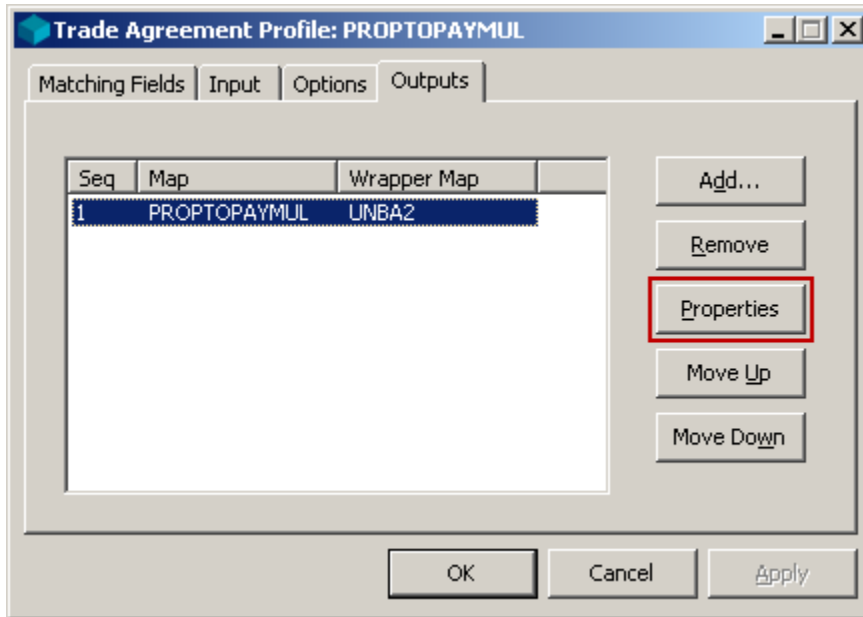
The following screen shows you page 3 of the partner definition for the recipient partner, BANK, which identifies the appropriate trade agreement profile, PROPTOPAYMUL.



The following screen shows you the trade agreement profile, PROPTOPAYMUL, for the incoming document, PML. The summary document, which is being generated here, is defined as part of the translation profile, on the next tab.


The screenshot shows a dialog box titled "Trade Agreement Profile: PROPTOPAYMUL". It has three tabs: "Matching Fields", "Input", and "Outputs". The "Matching Fields" tab is active. Under "Identified Standard:", there is a dropdown menu showing "CUSTOMER". Under "Input Field Values:", there are five text input fields: "Doc Id" (containing "PML"), "Version", "Release", "Agency", and "Assoc". At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

The **Outputs** tab of the Trade Agreement Profile window, specifies the output that identifies the security user exit, **BCEHash**, defined on the **Summary Documents** tab of the Trade Agreement Output Properties window. To view the properties window you select the **Properties** button. This user exit will be called to calculate the hash value of the segments in the outgoing document, PAYMUL. The summary document map, AUTACK, will call a security user exit to generate a digital signature based on the hash value, and will place the signature in the AUTACK message.

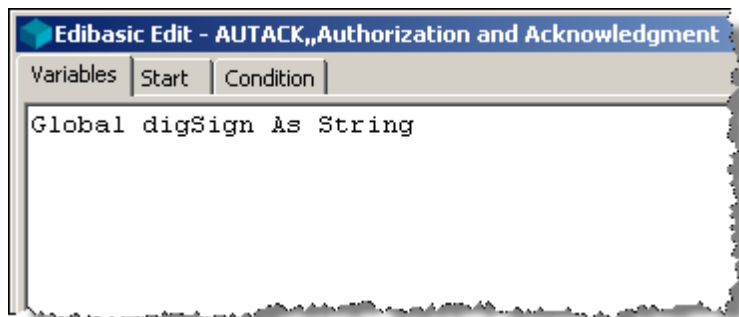


Specifying the User Exit to Generate the Digital Signature

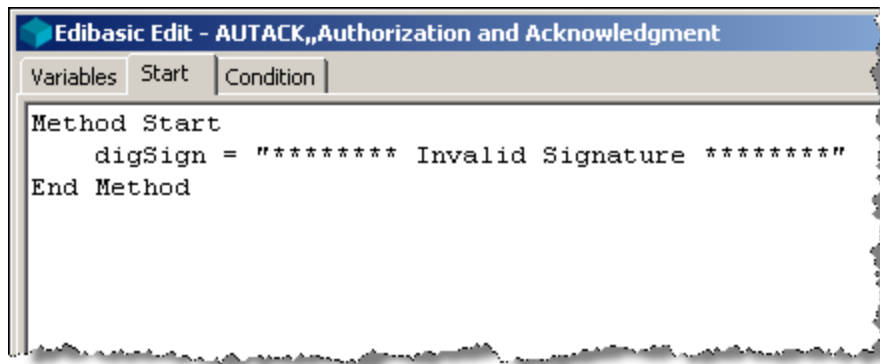
For the AUTACK map, you write an Edibasic method to call the Edibasic user exit that uses a public key and the hash value to generate a digital signature. Place the signature in the AUTACK message that will accompany the acknowledgment. Note that the AUTACK map is the same one discussed previously that the bank used to secure the CONTRL document. Note that the user exit saves the generated digital signature in that global variable, **digSign**. The following screen shows you the variable defined at the

document level. You can review this code by selecting the **Document Methods and Variables** button  next to the destination document header.

The variable is declared on the **Variables** tab.



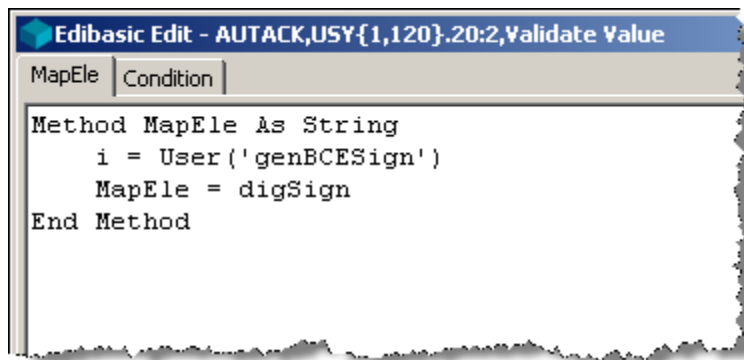
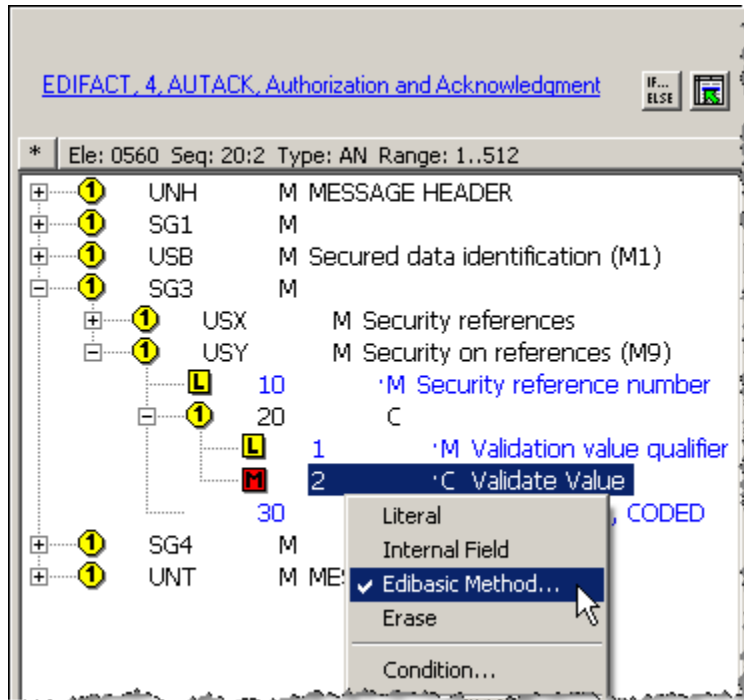
Note that the value is initialized on the Start tab.



The following screens show you the call to the user exit. You can also select the **Print Preview** button



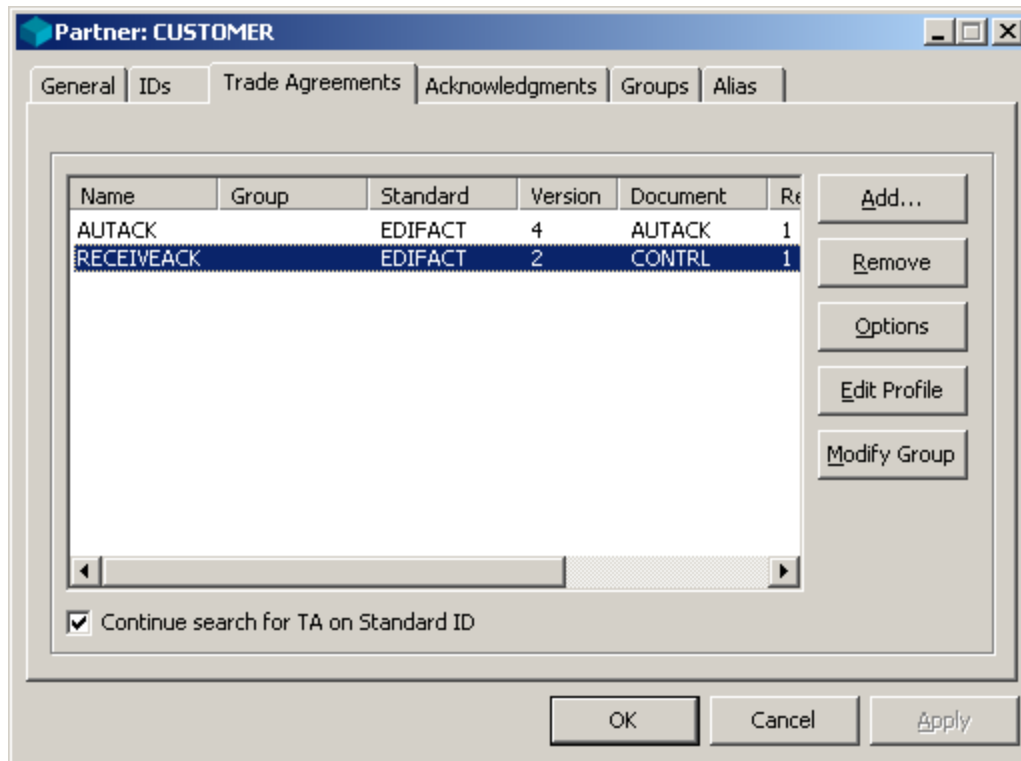
from the toolbar to view the code.



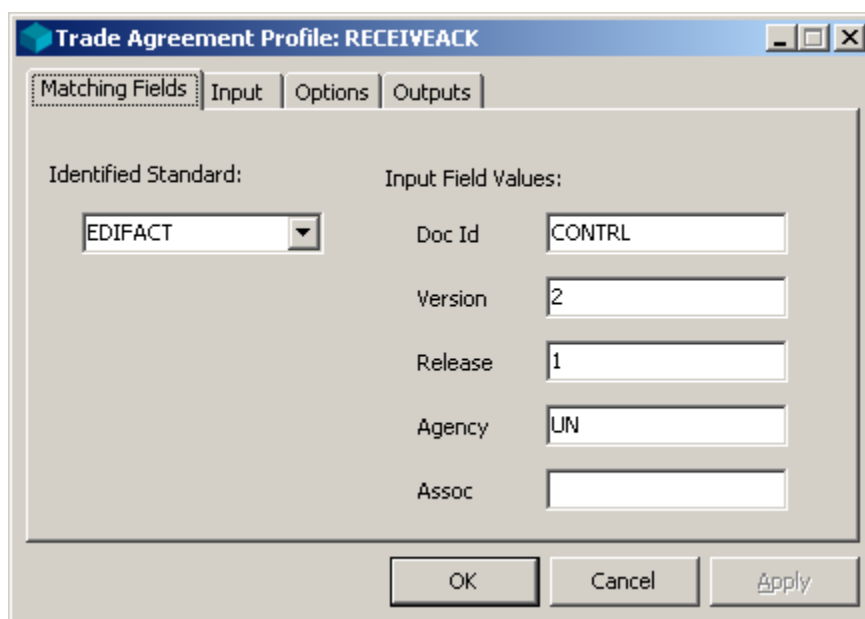
Specifying the Security Document and User Exit for Incoming Acknowledgment

For the recipient partner, configure the trade agreement profile for the incoming acknowledgment to identify the security user exit and the security document, AUTACK.

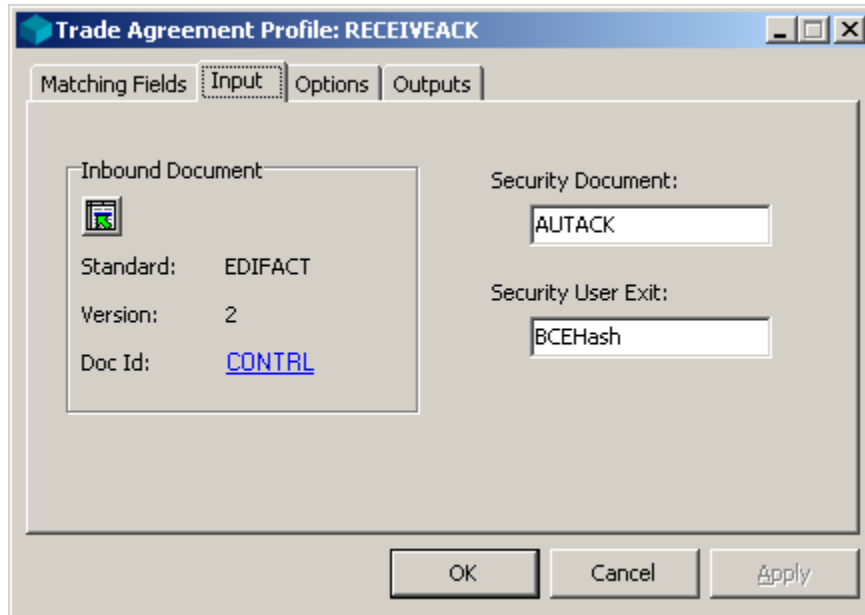
The following screen identifies the appropriate trade agreement profiles, RECEIVEACK and AUTACK, for the recipient partner, CUSTOMER.



The following screen shows you the trade agreement profile for the incoming acknowledgment, RECEIVEACK.



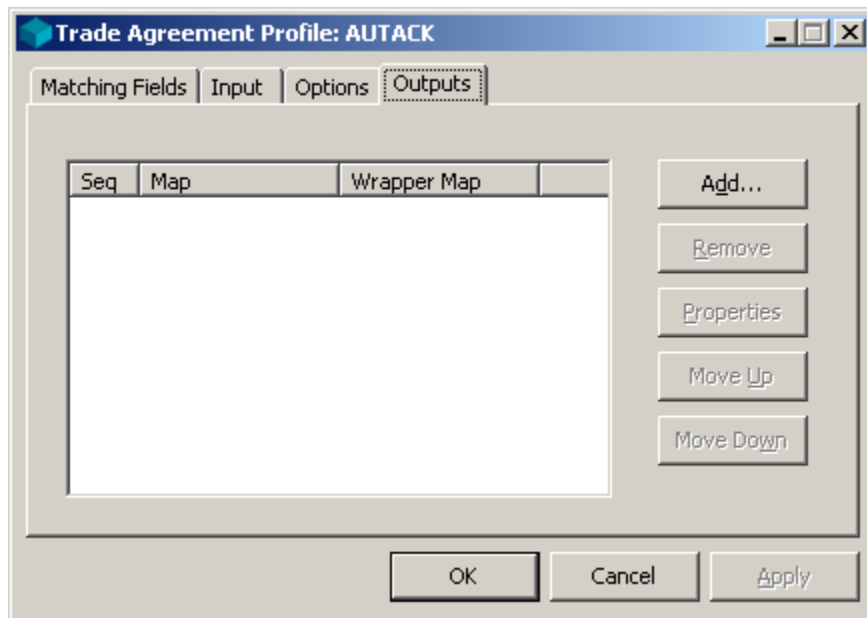
On the Input page it identifies the security document, AUTACK, that should accompany the CONTRL. It also identifies the security user exit, **BCEHash**, that will be called to calculate the hash value.



The following screens show you the trade agreement, AUTACK, that is received with the CONTRL. The TRM must parse the contents to be able to execute the Edibasic user exit that validates the digital signature. To accomplish this, you could select either **Validate Contents** or **Translate** from the **Options** tab. However, since you do not need any output from the AUTACK, you can select **Translate** and not put anything on the list on the **Outputs** tab. If you select **Validate Contents**, the TRM generates an exact copy of the input for output routing.

The screenshot shows the 'Trade Agreement Profile: AUTACK' dialog box with the 'Input' tab selected. The 'Inbound Document' section contains a document icon, 'Standard: EDIFACT', 'Version: 4', and 'Doc Id: AUTACK'. The 'Security Document' and 'Security User Exit' sections each have an empty text input field. At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

The screenshot shows the 'Trade Agreement Profile: AUTACK' dialog box with the 'Options' tab selected. The 'Action' section has radio buttons for 'Route File', 'Route File w/o Header', 'Validate Wrappers', 'Validate Contents', and 'Translate' (which is selected). The 'Error Action' section has radio buttons for 'Accept' and 'Reject' (which is selected). At the bottom are 'OK', 'Cancel', and 'Apply' buttons.



Specifying the User Exit to Validate the Digital Signature

For the AUTACK document definition, write a validation method to call the Edibasic user exit that validates the digital signature. The user exit validates the digital signature, returning accept or reject status to the TRM. Note that this is the same method used by the recipient, BANK, to validate its PAYMUL document. The following screens show you the code, but you can also select the **Print Preview** button



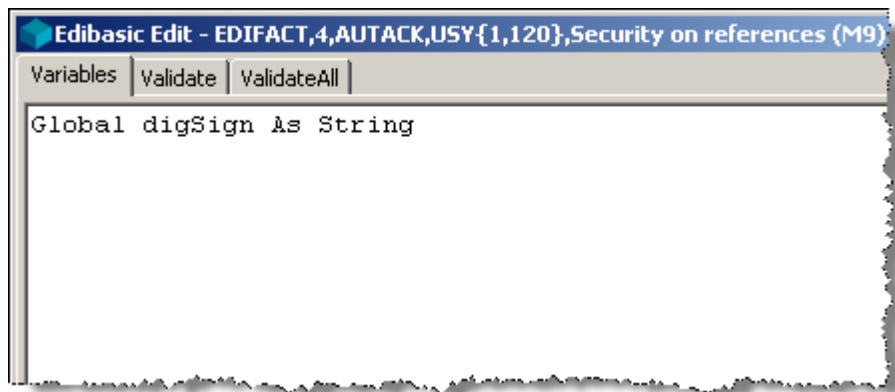
from the toolbar to see the Edibasic code associated with the **Validate Method**.

Document: EDIFACT, 4, AUTACK

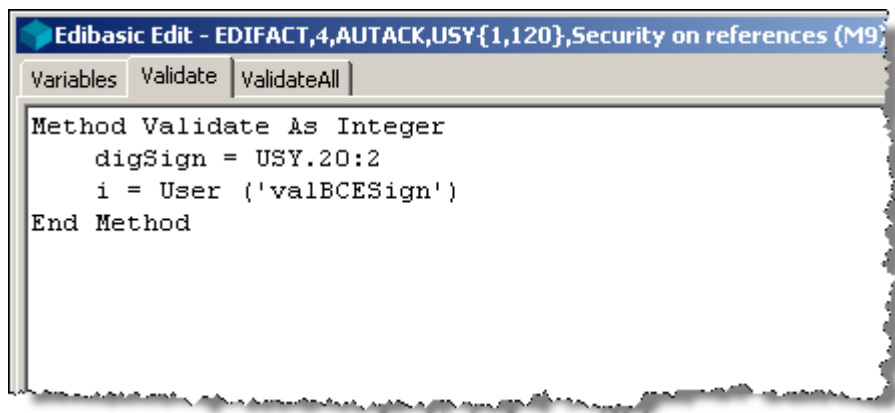
Description: Function:

L/S	Level	Area	Seq	Tag	Description	Rqmt	Max O
S	0	1	10	UNH	MESSAGE HEADER	M	
L	0	1	20	SG1	Loop	M	
S	1	1	30	USH	SECURITY HEADER	M	
S	1	1	40	USA	Security algorithm	C	
L	1	1	50	SG2	Loop	C	
S	2	1	60	USC	Certificate to convey public key and the credentials of its own	M	
S	2	1	70	USA	Security algorithm	C	
S	2	1	80	USR	Security Result	C	
S	0	1	90	USB	Secured data identification (M1)	M	
L	0	1	100	SG3	Loop	M	9
S	1	1	110	USX	Security references	M	
S	1	1	120	USY	Security on references (M9)	M	
L	0	1	130	SG4	Loop	M	
S	1	1	140	UST	Security trailer (M1)	M	
S	1	1	150	USR	Security Result	C	
S	0	1	160	UNT	MESSAGE TRAILER	M	

- Variables
- Validate
- ValidateAll
- Ack Wrapper Level
- Ack Document Level
- Insert Row
- Delete Row



```
Edibasic Edit - EDIFACT,4,AUTACK,USY{1,120},Security on references (M9)
Variables Validate ValidateAll
Global digSign As String
```



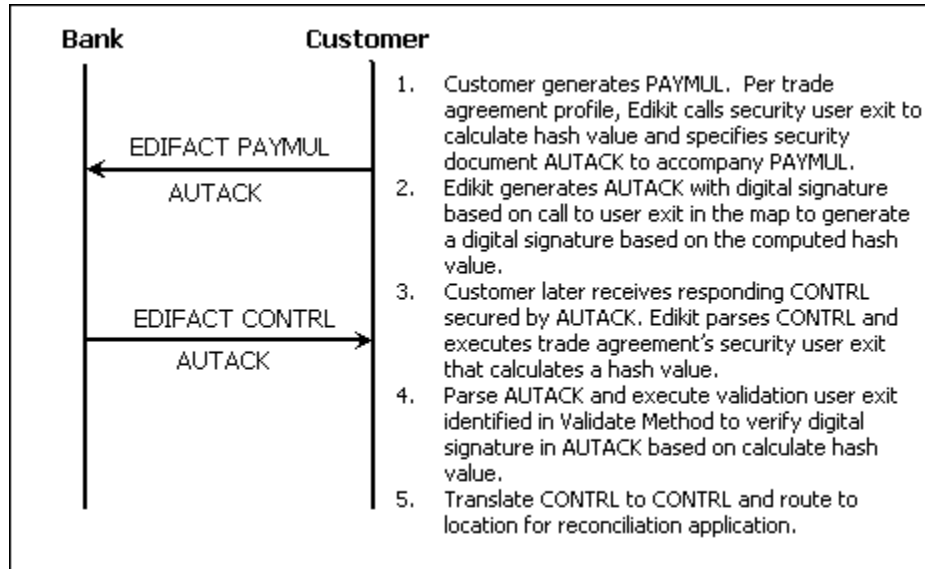
```
Edibasic Edit - EDIFACT,4,AUTACK,USY{1,120},Security on references (M9)
Variables Validate ValidateAll
Method Validate As Integer
    digSign = USY.20:2
    i = User ('valBCESign')
End Method
```

Example Input Files

Two sample input files have been included that reflect this type of exchange as explained in the following table. Refer to the **README** file included with the examples for more information.

Input file	Description
indata.txt	Processing includes proprietary payment information that generates a PAYMUL secured with an AUTACK.
incontrl.txt	Processing includes normal incoming CONTRL secured with an AUTACK message.

The following diagram reflects the behavior of the examples included in the AUTACK subdirectory. It shows the document exchange between a customer and the bank, from the viewpoint of the customer.



The following definitions are used to support the outgoing EDIFACT PAYMUL message secured with an AUTACK message:

Definition Type	Definition Names	Description
Wrappers	CUSTWRAP	Used to parse customer wrapper.
	UNBA	Used to generate EDIFACT PAYMUL wrapper.
Documents	AUTACK	Used to generate EDIFACT AUTACK message.
	PAYMUL	Used to generate EDIFACT PAYMUL message.
	PROPPML	Used to parse proprietary payment document.
Maps	AUTACK	Used to generate AUTACK.
	PROPTOPAYMUL	Used to generate EDIFACT PAYMUL.
	UNBA2	Used to generate EDIFACT wrapper.
Trade Agreement Profiles	PROPTOPAYMUL	Used to convert the proprietary payment to EDIFACT PAYMUL secured by AUTACK.
Standard Identification Locations and wrappers	CUSTOMER	CUSTOMER, 1, CUSTWRAP. Used to identify incoming proprietary payment.
	EDIFACT	EDIFACT, 2, UNBA. Used to identify incoming CONTRL.

Definition Type	Definition Names	Description
Partner Profiles	CUSTOMER	As sender of PAYMUL, receives responding EDIFACT CONTRL, secured with an AUTACK.
	BANK	Receives EDIFACT PAYMUL, secured with AUTACK.

The CONTRL message is received when the BANK trading partner returns it to the CUSTOMER after the BANK has processed the PAYMUL interchange. It uses some of the configurations listed for the proprietary payment processing, but for different purposes, and some additional configurations, which are all listed here:

Definition Type	Definition Names	Description
Wrappers	UNBA	Used to parse EDIFACT CONTRL wrapper.
Documents	AUTACK	Used to parse EDIFACT AUTACK message.
	CONTRL	Used to parse EDIFACT CONTRL message.
Maps	UNBA2	Used to generate EDIFACT wrapper.
	RCNCL	Used to copy CONTRL to CONTRL for reconciliation in CUSTOMER system.
Trade Agreement Profiles	RECEIVEACK	Trade agreement to receive and translate CONTRL documents.
Standard Identification Locations and wrappers	EDIFACT	EDIFACT,2,UNBA. Used to identify incoming CONTRL.
Partner Profiles	CUSTOMER	Receives EDIFACT CONTRL, secured with an AUTACK.
	BANK	Sends EDIFACT CONTRL, secured with AUTACK.
	RECONCILE-APP	Location where CONTRL messages are routed for reconciliation processing

Extensible Markup Language (XML) Support

This product provides integrated support for translation between the XML formats and other supported formats.

Overview

You may create definitions in MW Translator to represent XML data in one of two ways:

- Automatically from a separate Document Type Definition (DTD) file using the **DTD2TRN** utility or from a schema (XSD) using the **XSD2TRN** utility
- Manually from the Document, Segment and Element windows (composites are not used for XML). To generate XML output, only the data element values and attribute values need be mapped. When it generates XML data, the TRM creates the XML start and end tags and attribute names automatically

Validation of XML Data

MW Translator validates inbound XML data first using the pre-processing user exit, trmxml, and again in the MW Translator parser as part of its compliance checking.

- Validation by the XML parser in the pre-processing user exit occurs against a valid schema or dtd reference in the input xml data. There is no XML validation without such a reference.

CAUTION: When there are XML Xerces validation errors, not just warnings, the remaining validation and processing will fail, and the name of the root element will not be stored in the Data Element Store (DES). This will cause MW Translator error messages such as, "Invalid wrapper segment" and eventually, "Unable to find segments in the document."

- The MW Translator parser checks for compliance against the standard definitions once the file has returned to mainstream processing from the user exit and before it is mapped to the outbound data.

NOTE: There is no compliance check against the requirement designator when the value is **M** (mandatory), because MW Translator only does compliance checking for definitions that are found in the data. Of course, if they exist in the data, they will always pass the mandatory requirement check. Although the standards definitions provide the ability to validate the XML data, we recommend that the primary validation be done in the XML parser within the user exit against a DTD or XSD file. This implies that the definitions in MW Translator should be as open as possible.

MW Translator validates the output data when specified in the Trade Agreement Output Options. For more information, refer to the topic, *Validating Data During Generation* (on page 328). This process only validates that the data conforms to the MW Translator definitions.

To validate the XML format of the generated data, you have a few options, among them:

- Call an external validation program from a *post-processing user exit* (on page 329)
- Initiate an external validation program from a Custom Processing location within MessageWay
- Use an XML validation program outside of MW Translator and MessageWay

Example XML Translations

We provide two examples of XML translations:

- XML to X12
- X12 to XML

You can run an XML sample translation by importing the TRN file from the appropriate directory within the `\Workbench\Examples\xml` subdirectory. For further instructions, refer to the **README** file in the associated directory.

Creating MW Translator Definitions for XML

The basic steps to create MW Translator definitions for XML are as follows:

- 1 Create the definitions for the XML document:
 - *Use the utilities DTD2TRN or XSD2TRN* (on page 369) to convert an XML document type definition file (DTD) or a schema file (XSD) to an MW Translator transfer file (TRN)
 - or –
 - *Enter the definitions manually in MW Translator* (on page 373)
- 2 *When the input data contains namespaces, define those in MW Translator* (on page 376)
- 3 *Copy the appropriate wrapper definition from the XML examples to your XML standard* (on page 379):
- 4 *For input, specify the mandatory user exit, trmxml, to parse the XML data.* (on page 381)
- 5 *Create additional configurations and test* (on page 382).

Defining a Document from an XML DTD or XSD

DTD2TRN is a DOS command-line program that converts XML external DTD files to MW Translator transfer files. **XSD2TRN** is a DOS command-line program that converts schema files (XSDs) to MW Translator transfer files. Once a transfer file is created, you may import it into the MW Translator Workbench. The definitions created may be used for either parsing or generating XML documents in MW Translator.

IMPORTANT: Multiple file names and wild cards may be used to process many DTD files at once. In this case, all documents will belong to the same MW Translator standard version. This is the preferred way of importing multiple documents into the same standard version.

If related DTD files are processed individually and are imported to the same standard version, then there is a good chance that segment tag collisions will result. Collision occurs when the same segment tag is used for two different XML elements. Typically, this will break the definitions imported first. You can avoid segment tag collision by processing the files together as a group in the same DTD2TRN command. This is less of a problem with XSD files, because schemas typically use namespace references to avoid such collisions.

Limitations

The **XSD2TRN** and **DTD2TRN** utilities have the following constraint:

- The **XSD2TRN** and **DTD2TRN** utilities only support files that are encoded as UTF-8. If you have a DTD or XSD that is encoded differently, such as with UTF-16, you must first convert it to UTF-8 before you use one of these utilities.
- There must be no recursive definitions, as in the following incorrect example:

```
<!ELEMENT A (B,C)>
<!ELEMENT B (A,D)>
```

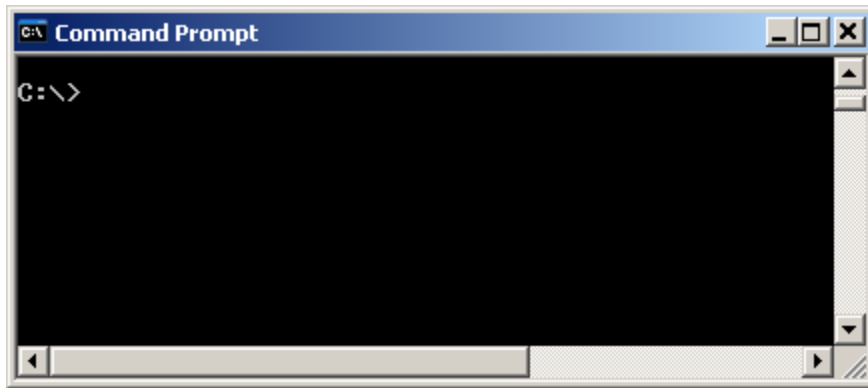
The **DTD2TRN** Utility has the following constraints:

- The DTD must be in its own file without any XML data, and then referenced within the XML file using a DOCTYPE statement.

Command Line Syntax

IMPORTANT: The **XSD2TRN** and **DTD2TRN** utilities only support files that are encoded as UTF-8. If you have a DTD or XSD that is encoded differently, such as with UTF-16, you must first convert it to UTF-8 before you use one of these utilities.

First you must open a command prompt, which for Windows XP is from the **Start>Programs>Accessories** location.



Enter the command including any appropriate path information. The syntax for the **DTD2TRN** and **XSD2TRN** commands are the same, as follows:

```
[dtd2trn | xsd2trn] [options] file(s)
```

Option	Description
-s <i>std,ver,desc</i>	Defines standard, version and description for the MW Translator standard version that will be used for these XML definitions. If values are not present, the program will prompt for them.
-d <i>docID,desc</i>	Defines the document ID and description to be used for the XML document. This only applies if a single DTD or XSD file is provided. If multiple files are being processed or if this option is not present, then the DocID is derived from the root element tag (truncated to 8 characters). If multiple files are processed and duplicates exist, then the ending characters will be replaced with numeric sequences to ensure that each document has a unique ID. The description is the root element tag followed by ' -- XML document from <file name.dtd>'.
-r <i>root</i>	Defines the tag of the root element. There is no indication within an external DTD or XSD file which element is the root element for a document. If this option is not provided or if multiple documents are being processed, then the program determines the root element as being the first defined element that is defined directly in the supplied DTD file (as opposed to those elements defined within nested DTD files).

Option	Description
-o <i>outputfile</i>	This defines the output TRN file. If provided, all definitions (even when multiple input files are provided) are written into this single TRN file. If this is not provided, then the output file will be the same as the input file with the extension of .TRN and there will be a separate output file corresponding to each input.
- -help	Display the program syntax on the screen.

Examples of Commands

The following commands convert all of the DTD or XSD files in the current directory to transfer files (TRN) of the same name. The program will prompt for the standard, version, and description to use.

```
dtd2trn *.dtd
```

```
xsd2trn *.xsd
```

The following command converts all of the DTD or XSD files in the current directory that start with a **0** or **1** to a single TRN file (**oasis.trn**). All definitions will be part of a single MW Translator standard version - XML,OASIS.

```
dtd2trn -s "XML,OASIS,OASIS XML Definitions" -o oasis.trn 0*.dtd 1*.dtd
```

```
xsd2trn -s "XML,OASIS,OASIS XML Definitions" -o oasis.trn 0*.dtd 1*.xsd
```

The following commands convert the file **sample.dtd** or the **sample.xsd** to a file named **sample.trn**. A single document is created - XML,EXAMPLE,SAMPLE using **sample** as the root element. This assumes that **sample** is defined as an element in the DTD or XSD.

```
dtd2trn -s "XML,EXAMPLE,Sample XML" -d "SAMPLE,Sample document" -r sample
sample.dtd
```

```
xsd2trn -s "XML,EXAMPLE,Sample XML" -d "SAMPLE,Sample document" -r sample
sample.xsd
```



Importing Transfer File to Workbench

Once you have created the transfer file, you can use the **IMPORT command** (on page 503) from the **File** menu in the Workbench to import the definitions, which will have extensions of .trn. The import process will resolve tag collisions for segment and element names.

In the event that you import an element or segment that exists in the standard version definitions, you will receive a prompt to overlay the definitions or not. If you know you want to overlay the definition, you can do so, or if you are not sure, do not overlay the definition. If you make a note of which definitions you do not want to overlay, you can reissue the import command. You can then select the individual definitions you want to rename, and import only those. For more information, refer to the topic, *Importing Configurations* (on page 416).

Completing and Reviewing MW Translator Definitions

When you import the TRN file, the utility creates a standard, version and document, as well as segments and elements. It also creates a basic data type for the Standard Version Profile. There are some additional steps you may need to perform to complete your definitions. To review these definitions, proceed as follows:

- 1 In the left pane, navigate to the XML standard and version for the document you just imported.
- 2 Check that the category for the standard is **XML**:
 - a) In the right pane, double-click **Standard Version Profile** for the appropriate standard and version.

 - b) Select the **Properties** button, , to view the **Category** value, which should be set to **XML**.
- 3 If you are using data types, review and *add any necessary data types* (on page 119) in the Standard Version Profile.
- 4 *You may have to define namespace in MW Translator* (on page 376):
 - To access namespaces in the input.
 - To create namespace declarations in the output, although namespaces may also be mapped
- 5 To verify the structure is correct, open the xml document you just imported and check the following:
 - Each XML element definition should have a corresponding MW Translator segment
 - XML elements that contain nested elements should be defined as MW Translator loops, and the first segment of the loop should have a category of blank or **XML**.

NOTE: Some XML schemas, notably those from X12, use extensive nesting. The loops might be several levels deep, and unlike traditional X12 structures, loops may be nested inside loops without any intervening segments. Therefore, you may notice empty segments in your structure. Empty segments are not allowed for traditional, non-XML, X12 and EDIFACT structural definitions.

- 6 Review the **Max Repeats** and requirement designator columns for the loops and segments.

NOTE: The requirement designators, Mandatory, Optional, Conditional, are not validated for input, but will be validated for output when you have configured your *output to be validated* (on page 328).

- 7 Double-click a segment to review the elements within the segments.
- For elements that use namespaces, the MW Translator element description should be of the form *namespace prefix : element tag*, for example, `po:Ship2`.
- 8 If you are going to generate xml data, double-click each element and select the **Properties** button,



, to view the **Category** value, or you can also *print a document report* (on page 571). For additional information about how MW Translator generates XML, refer to the topic, *How the TRM generates XML*. (on page 384)

NOTE: Within a segment definition, there should only be XML element data, category **XML** (**X** on the report), category **XML Element** (**XE** on the report) or **XML CDATA** (**XC** on the report), and these elements should be last, appearing after all elements of category **XML Attribut** (**XA** on the report).

- a) MW Translator elements that represent XML attributes should be at the beginning of the segment, and each should have a category of **XML Attribut**. The element ID is arbitrary, but the element description should be the same as the attribute name.
 - b) If the XML element contains data, an MW Translator element should be defined, and for the *category option* (on page 596):
 - Set the category to **XML Element**, when this is the lowest level where no elements in the segment contain nested elements or attributes, and the element description will be used to generate the XML element name
 - or –
 - Set the category to **XML** or blank so the data will become part of the parent within the parent's start and end tags, without tags of its own
 - or –
 - Set the category to **XML CDATA**, which will result in the generated data being enclosed in a CDATA section, within the parent's start and end tags
-

IMPORTANT: You should define only one element with a category of **XML CDATA**, **XML** or blank for any segment of category **XML**.

Defining an XML Document Manually

To define an XML document manually, you must define its standard and version, and any data types associated with the document. Then you must define the structure of the document. It might also be helpful to review the information in the topic, *Defining a Document from an XML DTD*. (on page 369)

Defining an XML Standard and Version

You can define an XML document by following these steps.

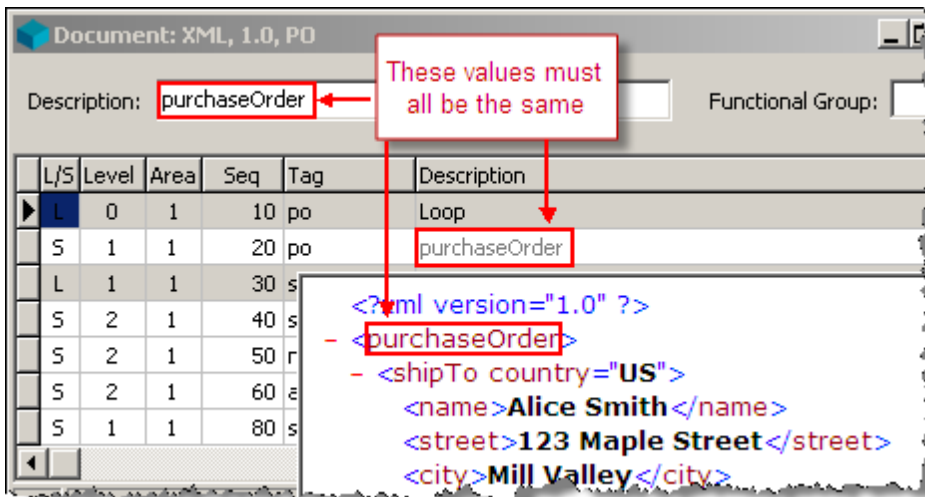
- 1 **Create your XML standard and version** (on page 118), and set the category on the Standard Version Properties window to **XML**.
- 2 **Add any data types** (on page 119) you expect to use to the Standard Version window.

Defining the Structure of the XML Document

Create a document to model the XML document with appropriate nesting of loops. For more specific information about how the TRM generates XML based on these definitions, refer to the topic, *How the TRM generates XML* (on page 384).

NOTE: The nested hierarchy of XML elements should be defined in MW Translator as nested loops and segments. The MW Translator loop IDs and segment tags are arbitrary. It is the description of the segments, and typically the elements as well, that is important, since it must match the XML element tag.

- 1 **Create a document** (on page 568), and for the description, type the name of the XML root element or document node.



- 2 For the XML root element:
 - a) Create an outermost loop in MW Translator with one segment
 - b) Type the name of the XML root element as the description of the MW Translator segment, which must be exact, since it is case-sensitive. The segment tag may be any unique 6-character value, but it is not significant for XML standards.
 - c) Create MW Translator elements for the segment to represent each of the attributes that may be present in the root declaration.
- 3 Ignore any XML comments. They should not be defined in MW Translator.

IMPORTANT: XML comments are ignored during parsing, and they are not stored in the Data Element Store. XML comments cannot be mapped to the output. This is in compliance with the W3C recommendation for the treatment of XML comments.

- 4 For each XML element definition, add an MW Translator segment to the document.
 - a) The segment tag is arbitrary, since it is not used to generate XML, but it must be unique for MW Translator.
 - b) For XML elements defined as loops (because they contain nested elements), the XML element tag must be the same as the description of the first MW Translator segment in the loop. Type the segment description exactly to match the XML element tag. This value is case-sensitive.
 - c) For nested XML elements, the XML element tag must be the same as the description of non-initial MW Translator segments in the loop/group.
 - d) For MW Translator elements that use namespaces, the MW Translator element description should be of the form *namespace prefix:element tag*, for example `po:shipTo`.
 - e) Add this segment to the document.
- 5 If the XML element contains nested elements, then add a loop containing the previously defined segment. The description of the first MW Translator segment in the loop must be exactly the same as the XML element tag.

NOTE: All nested elements should be defined as nested segments within this loop. The first segment in the loop represents the outer XML element, and might not have any elements defined, while subsequent segments represent nested XML elements.

- 6 For each XML attribute defined within the XML element, in MW Translator add an element to the segment just defined.
 - a) The element ID is arbitrary, since it is not used to generate XML, but it must be unique for MW Translator.
 - b) Set the element description to the XML attribute name.
 - c) Set the element category to **XML Attribut**.
- 7 If the XML element contains data, then add an element to the segment, and for the *category option* (on page 596), which is used when you generate XML data, do one of the following:
 - Set the category to **XML Element**, when this is the lowest level where no elements in the segment contain nested elements or attributes, and set the element description to the XML element name
– or –
 - Leave the category blank or **XML**, in which case the element ID and description are arbitrary, and the data will be enclosed in the start and end tags created from the description of the parent segment
– or –

- Set the category to **XML CDATA**, which will result in the generated data being enclosed in a CDATA section, which will be enclosed in the start and end tags created from the description of the parent segment
 - or –
- Set the category to **Fixed**, which will not generate any XML tags, so you can map an XML comment, for example.

IMPORTANT: There should only be one element of category **XML**, blank or **XML CDATA** within a segment definition, and this element should be last, appearing after all elements of category **XML Attribut**.

Specifying Namespaces

Namespaces are a label in a Uniform Resource Identifier (URI) format. They point to schemas that are used to avoid collisions of XML tags, where the same tag within different schemas may have different parameters associated with it, such as different data types or lengths. Namespaces are declared within the start tag of the XML root element or document node. For purposes of shorthand, prefixes may be used in the in the root element to reference the namespace, using the syntax, `xmlns:prefix = "URI"`. Then, further in the data, a particular element will use the prefix as part of its tag, using the syntax, `prefix:element tag`.

The URI may or may not be a valid Uniform Resource Locator (URL) or Uniform Resource Name (URN). When a namespace is used, but it is not available, you will receive one or more warnings. It may be unavailable, for example, because it is an invalid URN or URL, or you may not have security access to a valid URL.

CAUTION: When you do not need namespaces to avoid collision of XML tags, you should not define them in MW Translator. You will receive warnings when namespaces appear in the input but they are not defined in MW Translator. These warnings will not stop translation. This way, if a partner sends you data that may or may not contain a namespace, you will only receive a warning. Alternatively, if you do define namespaces in MW Translator, but you do not receive them in the data, your translation will probably abort.


When namespaces are used in the input data and are required to avoid collisions, you must specify those in MW Translator. You typically define namespaces on the first segment of the document, which represents the XML root element. Namespaces may be defined on other segments, but they must be defined before they are referenced by an element.

To Define a Namespace to Parse Input

To define a namespace in MW Translator on the first segment of the document, proceed as follows:

- 1 In the left pane, select **Segments**, and then in the right pane, double-click the first segment in the document, which should represent the XML root element.



- 2 Click the **Properties** button,  To, and from the **Category** list, select **XML**.
- 3 The **XML** tab appears.
- 4 On the **XML** tab, type each namespace, one per line, using the format, `xmlns:prefix="URI"`. For example, you might type:

```
xmlns:po="http://myco.com/schema/purchaseOrder"
```
- 5 Repeat steps 1-4 for each namespace in the input data, and type each namespace on a separate line.

To Generate a Namespace for Output

To generate a namespace in XML output, you can use one of two methods, or a combination of these:

- *Define the namespace on the XML tab of the segment* (on page 376)

NOTE: Any namespaces defined on the **XML** tab of an MW Translator segment are automatically generated as an attribute of the XML element. These are the same namespaces defined for the input.

– or –

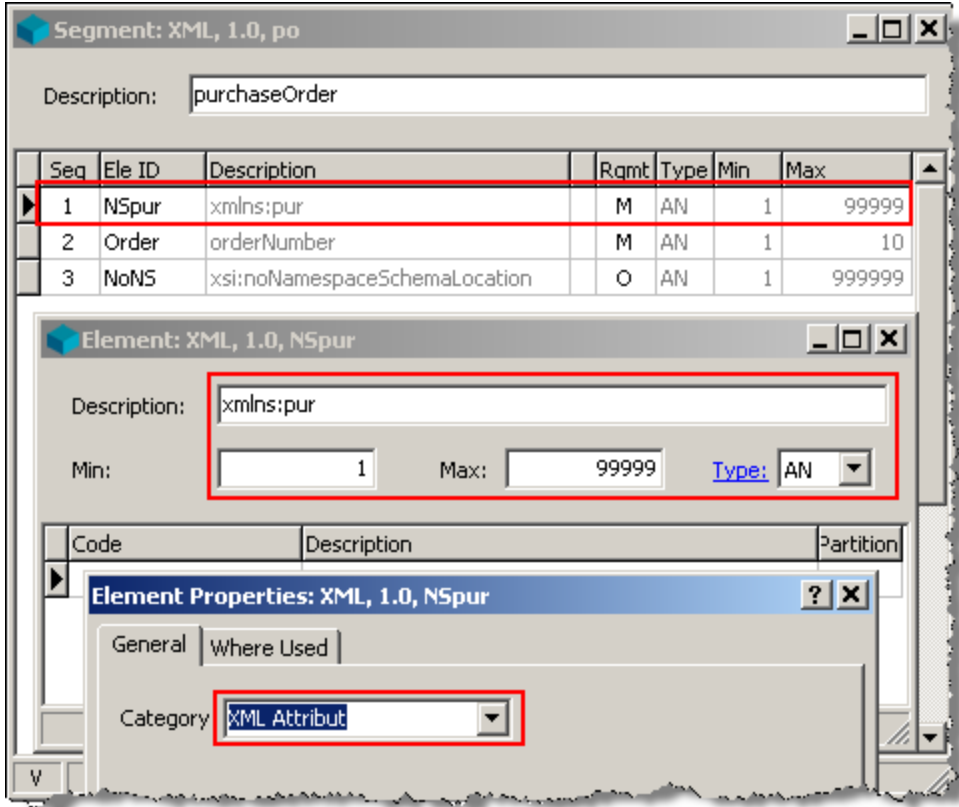
- Use Edibasic mapping

NOTE: In the event that you do not want to define namespaces for the input, you can use Edibasic to map the namespaces to the output. You can also use this method when there are namespaces defined on XML tabs, to add more namespace declarations, perhaps on different segments.

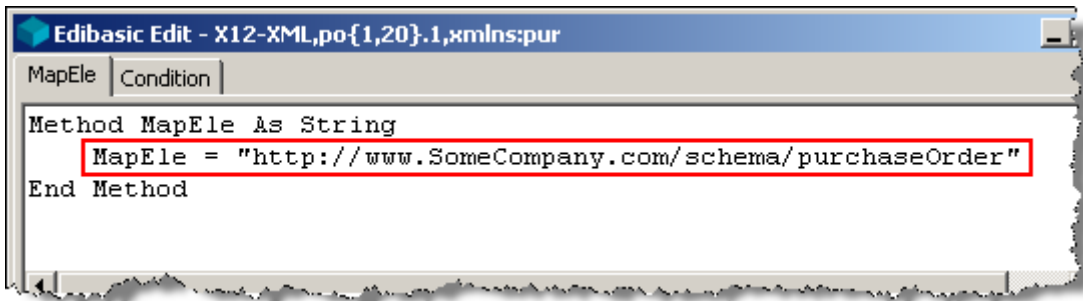
Typically, you would specify the namespace on the first segment of the document, which represents the XML root element. To use Edibasic to map namespaces to the output, proceed as follows:

- 1 *Create an element to represent the namespace* (on page 590).
 - a) The Element ID should be unique for MW Translator, but it is not used in the generated XML.
 - b) The description of the element will be used to generate what precedes the equal sign, so it should be of the form, `namespace:prefix`. If you are not using a prefix, leave that part out.
 - c) From the Element Properties window, select the Category, **XML Attribut**, to specify that this is to be generated as an attribute.

- 2 **Insert the element in the appropriate segment** (on page 628), typically the first segment of the document, as shown in this example:



- 3 Open the map that generates the XML output document.
- 4 **Reselect the document for the output** (on page 608), which will display the new element in the first segment.
- 5 **On the new element, use Edibasic to add the URI for the namespace to the MapEle tab** (on page 615). Enclose the URI in single or double quotation marks, as shown in the following example.



- 6 Select **Generate All** and run a test. This will produce output similar to the following:



```
AFDNL4.txt - Notepad
File Edit Format View Help
<?xml version="1.0" ?>
<purchaseOrder
xmlns:pur="http://www.SomeCompany.com/schema/purchaseOrder"
orderNumber="PO12345">
```

Defining the XML Wrapper

MW Translator treats XML input and output slightly differently.

For input in general, MW Translator requires at least a null input wrapper. However, for XML input, MW Translator requires a special null wrapper, pre-defined for your convenience, to store the root tag as the document ID. The XML declaration is removed by the trmxml user exit before it parses the data. Therefore, the XML declaration is not part of the wrapper definition.

For XML output, a separate pre-defined wrapper provides a definition for the XML declaration line, which a wrapper map creates. Therefore, the XML declaration is defined as an element in the output wrapper.

To Copy the XML Null Input Wrapper

The XML input wrapper uses a special wrapper definition whose purpose is to pass the root element as the document ID to the MW Translator XML parser, which is part of the trmxml user exit. You can use the same wrapper definition used in the XML-X12 example.

- 1 If you have not done so, import the XML-X12 example.
 - a) From the **File** menu, click **Import**.
The **Import Definitions From** dialog box appears.
 - b) Navigate to the location of the transfer files for the XML-12 example, **\\Workbench\\Examples\\xml\\XML-X12**, and select **xml.trn** and click **Open**.
The Import Definitions window appears.
 - c) Click **Import** to import all of the examples definitions (documents, wrappers, maps, profiles, partners, etc).
- 2 Copy the XMLIN wrapper to your XML standard from the standard used by the MW Translator example.
 - a) In the left pane of Data Explorer, navigate to the wrapper definition in **Standards|XML|1.0|Wrappers** folder.

- b) In the right pane, right-click **XMLIN**, and from the menu click **Copy to**.
The **Enter the destination for copy** dialog box appears with your environment selected.
 - c) Check the box, **Rename Definition**, and click **OK**.
The **Rename** dialog box appears.
 - d) Type the names of your XML standard and version, and click **OK**. If they do not exist, they will be created.
- 3** For the **Contents** header in the wrapper definition, specify an existing document ID:
- a) In the left pane for the appropriate folder for your new standard and version, click **Wrapper**.
 - b) In the right pane, double-click **XMLIN**.
The Wrapper window appears.
 - c) In Contents header box, type the document ID of an existing document.

IMPORTANT: For XML, the document listed in the Contents header box does not need to match the document parsed at runtime, which will be specified on the **Input** tab of the Trade Agreement Properties window. The XML user exit supplies the XML tag of the root element that MW Translator stores in the Document ID field of the Data Element Store (DES). MW Translator compares this value with the description of the input document specified on each of the trade agreements on the trade agreement list to determine which trade agreement to use. This allows you to specify multiple trade agreements for a single location. Other types of input use a similar process to find a trade agreement, but they match the document ID stored in the DES against the Doc Id field specified on the Trade Agreement Properties window.

To Copy the XML Output Wrapper

The XML output wrapper is optional, but you must specify a wrapper. It provides the configuration for the XML declaration statement. You can use the same wrapper definition used in the X12-XML example.


- 1** If you have not done so, import the X12-XML example.
 - a) From the **File** menu, click **Import**.
The **Import Definitions From** dialog box appears.
 - b) Navigate to the location of the transfer files for the X12-XML example, **\\Workbench\\Examples\\xml\\X12-XML**, and select **xml.trn** and click **Open**.
The Import Definitions window appears.
 - c) Click **Import** to import all of the examples definitions (documents, wrappers, maps, profiles, partners, etc).
If you have imported the xml.trn from the XML-X12 example, some of the definitions will already exist. You can overlay these definitions.
- 2** Copy the XMLOUT wrapper to your XML standard from the standard used by the MW Translator example.

- a) In the left pane of Data Explorer, navigate to the wrapper definition in Standards\XML\1.0\Wrappers folder.
- b) In the right pane, right-click **XMLOUT**, and from the menu click **Copy to**.
The **Enter the destination for copy** dialog box appears with your environment selected.
- c) Check the box, **Rename Definition**, and click **OK**.
The **Rename** dialog box appears.
- d) Type the names of your XML standard and version, and click **OK**. If they do not exist, they will be created.


Specifying the User Exit to Parse Input

To validate and parse XML input, you must use the trmxml user exit. The pre-processing user exit to validate and parse xml data is called trmxml and is delivered with the Workbench and the MW Translator service with the MessageWay Server. To configure the user exit, proceed as follows:

1 For the Workbench:

- a) Click the **Options** button, , on the toolbar.
The Modify Options window appears.
- b) Click the **User Exits** tab and then the **Add** button to select the user exit, trmxml.dll, which should be in the \Workbench\bin directory.

2 For an MW Translator service in MessageWay, specify the location of the user exit within the MW Translator Operator program.

- a) Click the **Options** button, , on the toolbar.
The Options for MessageWay window appears.
- b) Click the **Server Options** tab and log on to MessageWay.
- c) Click the **User Exits** tab, and type the full path name of the trmxml file.
- d) Click **OK**.

Completing XML Configurations and Maps and Testing

XML document mapping is identical to mapping to other formats. You can drag and drop source data items to XML attribute or XML element definitions. Repeating loops and segments from the source document may be dragged to XML loops and segments in the destination in order to control repeating XML elements. Likewise, Edibasic methods may be defined for more advanced mapping. For a discussion of mapping strategies and testing, refer to the topic, *Map Development Guidelines* (on page 107).

For working examples of translations to and from XML, refer to the two examples in the subdirectory, \Workbench\Examples\xml in your installation directory. Load the .trn files into MW Translator to create the configurations and run tests using the supplied input data. You can then review the configurations as examples to follow.

Testing XML Input

If your output is also XML, refer to the topic, *Testing XML Output* (on page 384).

For XML input, proceed as follows:

- 1 Create a map to generate the output document.
- 2 *Define a trade agreement profile* (on page 222). There are *special considerations for inbound XML trade agreement profiles* (on page 383).
- 3 (Optional) *Create partner records for the output* (on page 243), if required.
- 4 *Create a location/Std ID record* (on page 297) to identify the inbound standard and wrapper. *There are some special considerations for inbound XML standards* (on page 383). Specify the trade agreement on the Standard Identification configuration.

NOTE: You could define default partners on the Standard Identification window, but this would limit the inbound documents that could use this location and wrapper to those specific partners, forcing you to create multiple source locations for each different recipient partner or partner pair.

- 5 Create test data.
 - If you validate against a schema (xsd) or document type definition (dtd) file, specify the appropriate location of the file.
- 6 *Test your translation* (on page 387).

Defining Trade Agreements for XML Input

There are some special considerations when you create a trade agreement for incoming XML data. On the Trade Agreement Profile window, you should note the following:

- On the **General** tab, the **Doc Id** should be blank. Any value here is ignored. All other entries should also be blank.
- On the **Input** tab, the document selected here **MUST** match the input. For XML, the description of the selected document must match the XML root element tag, which was stored in the Data Element Store as the Document ID.
- On the **Options** tab, you must select **Translate**. The other options are invalid for XML input data.
- On the **Outputs** tab, add one or more outputs, and specify the maps for the output wrapper and document on the Trade Agreement Output Options window.

Creating a Source Location to Identify Inbound XML

Since raw XML input is not constrained to place information in consistent positions, you will not be able to easily identify the input based on parsed data. Therefore, it is easier to create a separate source location for the XML input.

IMPORTANT: In the event that users send other types of data to the same location, the XML definition should be the last wrapper identified on the list for the location.

Follow these guidelines:

- 1** Create a source location for each different XML wrapper. For example, if you only use the XMLIN wrapper, you can use that for all XML documents within that standard and version that have unique root element tags.
- 2** On the Standard ID window:
 - a) On the **General** tab, you do not need to enter any information in the Matching Criteria box, so leave it blank.
 - b) On the **Partners** tab:
 - Make sure the Partner Definition Required boxes are NOT selected
 - Typically, you would not define default partners, which would limit the use of this Standard ID definition to those partners.
 - c) On the **Routing** tab, specify any default locations as you would for any other type of input.
 - d) On the **Trade Agreements** tab, select all trade agreements you have created for XML input within this XML standard and version. They must have unique XML root element tags, which must match the description of the MW Translator document and the first segment in the document. This is the document you selected on the **Input** tab of the Trade Agreement Properties window.
- 3** *To specify output partner IDs, an override location or miscellaneous settings for the output* (on page 881), from the **Trade Agreements** tab, click the **Options** button.

Testing XML Output

If your input is also XML, refer to the topic, *Testing XML Input* (on page 382).

To test XML output, you should proceed as follows:

- 1 Create a wrapper map
 - (Optional) To generate the XML declaration, copy the XML wrapper map, XML-WRAP, from the XML output example, X12-XML, to your XML standard.
 - or –
 - Specify a null wrapper
- 2 Create a map to generate the document. There are some special considerations to generate *XML namespaces* (on page 377) and XML comments.
- 3 *Define a trade agreement profile* (on page 222). If the inbound standard is also xml, there are *special considerations for inbound XML trade agreement profiles* (on page 383).
- 4 (Optional) *Create partner records* (on page 243).
- 5 *Create a location/Std ID record* (on page 297) to identify the inbound standard. If the inbound standard is also xml, *there are some special considerations for inbound XML standards* (on page 383).
- 6 Specify the trade agreement *on the Standard Identification, recipient partner or partner relationship*, (on page 20) according to your normal procedure. If your input is also XML, you should follow the guidelines in the topic, *Defining Trade Agreements for XML Input*. (on page 383)
- 7 Create test data.
- 8 *Test your translation* (on page 387).

How the TRM generates XML

The TRM generates XML in the same way as other formats - it traverses the document definitions executing mapping instructions at each node. Repetition and data values are controlled by the mapping instructions. Formatting, for segment tags, delimiters, XML element start and end tags, is controlled by the document definition. The following sections describe how each component of the XML document definitions generates part of the resulting XML output document.

Loop generation

For loops of category **XML**, the TRM generates a start and end tag based on the description of the first segment in the loop. Attributes are generated based on elements of category **XML Attribut** at the beginning of the first segment of the loop. Element data is generated based on an element of category **XML** (or blank) in the first segment of the loop. Nested XML elements are generated for each segment in the loop other than the first segment.

Segment generation

For segments of category **XML**, the TRM generates a start and end tag based on the description. Attributes are generated based on elements of category **XML Attribut** at the beginning of the segment. Element data is generated based on an element of category **XML** (or blank). Nested XML elements are generated for each element of category **XML Element**. If the element contains no data and no nested elements, then an empty element tag (with attributes if defined) is generated instead of separate start and end tags.

Element Generation (category of Fixed)

To create comment type elements, you should specify the MW Translator element category as **Fixed**. This does not generate tags. You use this category to generate comments, but you must also use Edibasic to format a proper XML comment.

Element generation (category XML Attribut)

For elements of category **XML Attribut**, the TRM generates the attribute name from the element description. The Attribute value is generated from the mapped value after replacing instances of **<**, **>**, **&** and **"** with **<**, **>**, **&** and **"**, respectively.

Element generation (category XML or blank)

For elements of category **XML** (or blank), the TRM generates the mapped data after replacing instances of **<**, **>**, and **&** with **<**, **>** and **&**, respectively. The element start and end tags should be generated using the XML tags as part of the parent segment, using the tags of the parent segment.

Element generation (category XML CDATA)

For elements of category **XML CDATA**, the TRM generates the mapped data prefixed by `<![CDATA[` and suffixed by `]]>`. The element start and end tags should be generated as part of the parent segment using the tags of the parent segment.

NOTE: You should only define one element of category **XML CDATA**, **XML**, or blank per segment of category **XML**.

Element generation (category XML Element)

For elements of category **XML Element**, the TRM generates the start and end tags from the element description. The TRM generates the element contents from the mapped data after replacing instances of `<`, `>`, and `&` with `<`, `>`, and `&`, respectively.

Limitations

XML support has the following limitations:

- There is no support for wide-character encoding.
- XML support is only provided for document definitions. Other format wrappers or a null wrapper must be used with XML documents.
- There is no direct support for generating the XML prolog, schema or a DTD. However, you can code these as literals in the map.
- There is no internal XML validation of output.

Testing Configurations

Information Required for Testing

For testing, at a minimum you must have the following entities for any type of work:

- Input test data
- Wrapper definitions to parse the incoming wrapper
- Standard ID profile to specify wrapper definitions to match the incoming data in order to know which wrapper definition to use to parse the data, and to specify any default attributes associated with processing the data
- Trade Agreement profile to identify what work to do

For contents validation and routing, you must have at a minimum the following additional configurations:

- Document definitions to parse the incoming document

For translation, you must have at a minimum the following additional configurations:

- Wrapper definitions for outbound wrapper
- Document definitions for outbound document(s)
- Map to generate outbound wrapper
- Map to generate outbound document

Optional configurations you might require for translation are:

- Cross-reference tables
- Acknowledgment profile
- Wrapper definitions for returned acknowledgment
- Acknowledgment definition(s)
- Map for acknowledgment wrapper
- Map for acknowledgment
- Partner profiles
- Partner relationship
- Group

Important Settings for Tests


There are some Workbench settings and some TRM settings that will help you control your testing.

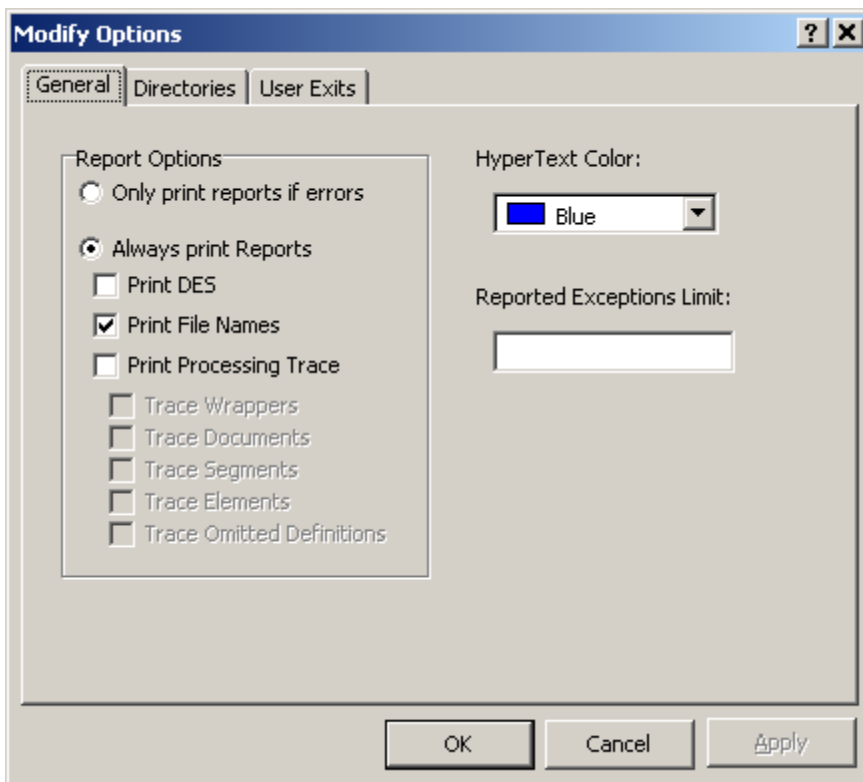
Setting Report Options

Before you generate any files, make sure the TRM will produce a processing report. The TRM produces processing reports by default only when there are errors during processing or when it encounters print statements in a map.

You can require that reports be produced as well as control the number of errors that will print, just in case you have too many repetitive ones. To always print a report, you select one of the options in the **Report Options** box.

To set the report options, perform the following steps:

- 1 Select the **Options** button  from the toolbar. The Modify Options window appears.



- 2 Within the **Report Options** box, select **Always Print Reports** and **Print File Names**.
- 3 Select **OK**.

Setting TRM Options

Until you are more familiar with the Workbench, we suggest that you generate output that is easy to read and that you save your reports and output.

For readability, you can require the Workbench to generate your test output data to display each segment on a separate line. This does not affect how the TRM generates the data in the production environment. To set this switch, select **Force Text** from the **Test** menu.

To save your output, make sure that the **Auto-delete Outputs** switch is not checked on the **Test** menu. By default, this switch is not selected. When the **Auto-delete Outputs** switch is checked on the **Test** menu, the Workbench deletes the files when you exit the Test window.

To save your reports, make sure you rename them after each test. The file names of processing reports are the same as the name of the input file used for testing. When you test using the same input file, the TRM overlays the old processing report with the new one.

Generating Text Files

You use the Workbench windows to enter and display your standards definitions, partnership information, and mapping instructions in relational databases. From this database information, you must generate text files to be used during EDI processing, which improves throughput, and allows for platform independence. When you generate text files, you generate potentially two different kinds: a Workbench file for testing and, for non-Windows systems, a target file for production processing.

You must generate text files for the following entities (the last 5 files are only required when you translate):

Generated Entity	Generate Command	Definitions Include
input wrapper(s)	All (include Documents), or specific document	segments, elements, composite elements, standard types, and delimiters if needed
input document(s)	All (include Documents), or specific document	segments, elements, composite elements, standard types
standard ID	All or Std ID File	all standards and versions, including delimiters for delimited standards, and their configured identification and default parameters
partner file	All or Partner File	all configured partner profiles
partner relationship file	All or Partner File	all configured partner relationships

Generated Entity	Generate Command	Definitions Include
profiles	All or Profile File	all configured trade agreements and acknowledgments
(translation only) output wrapper(s)	All (include wrappers), or specific wrapper	segments, elements, composite elements, standard types, and delimiters if needed
(translation only) output document(s)	All (include documents), or specific document	segments, elements, composite elements, standard types
(translation only) map for each output wrapper	All (include maps), or specific map	one map file for each output wrapper(s) (you can define more than one output per input)
(translation only) map for each output document	All (include maps), or specific map	one map file for each output document(s), (you can define more than one output per input), including acknowledgments
(translation only) Xref file, if you are doing any cross-referencing	All or Xref	all cross-reference tables

NOTE: The first time you create definitions on your system, use the **GENERATE>ALL** option from the Workbench window. This will assure that you have all the text files present to begin testing. Thereafter, if you make changes, you can regenerate only those files that changed.

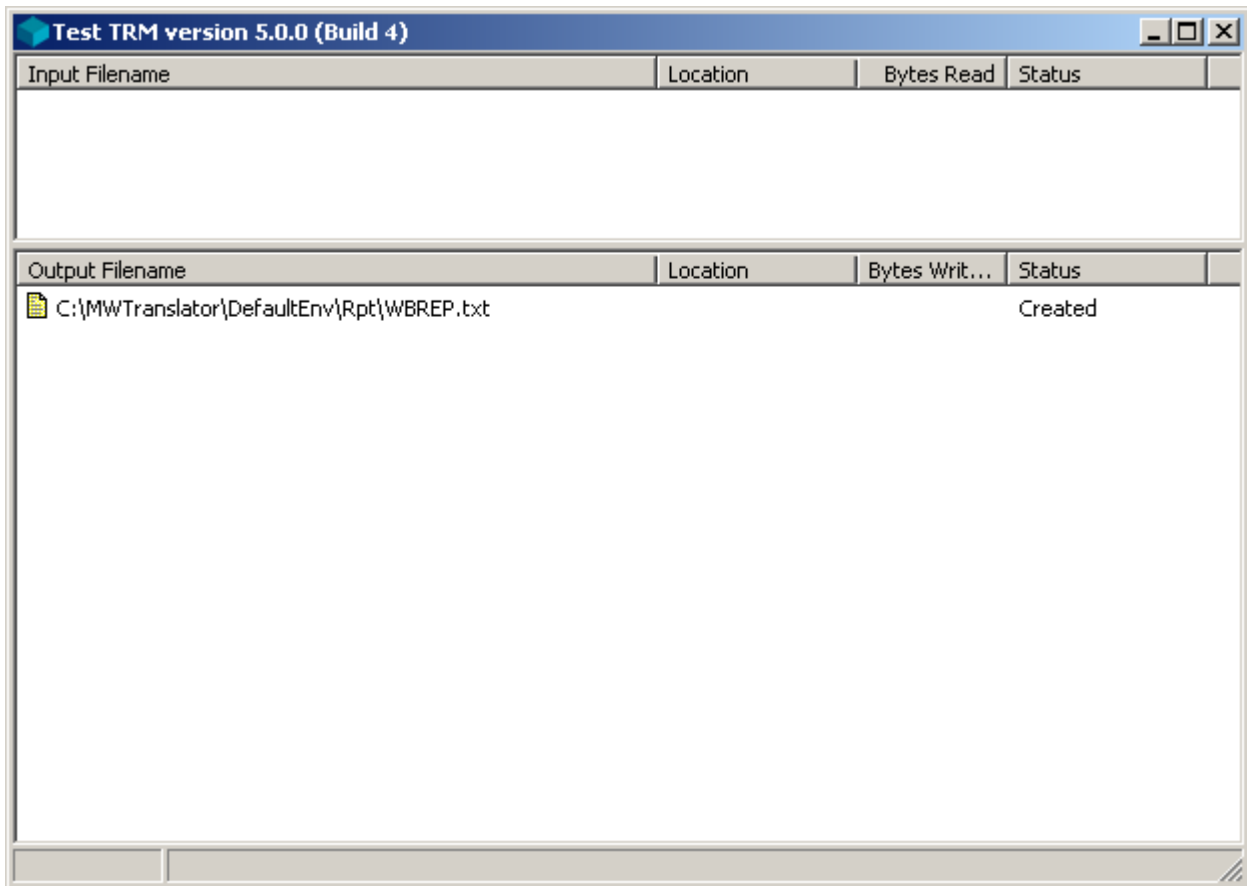
Running a Test

Testing includes creating test data, submitting the data for testing, then perhaps correcting mistakes and resubmitting the data until you process without errors. Some problems are a result of incorrect configurations. Your configurations determine what kind of processing occurs. If you are translating, some problems might be the result of an incorrect map.

To test your configurations on the Workbench, you perform the following steps:

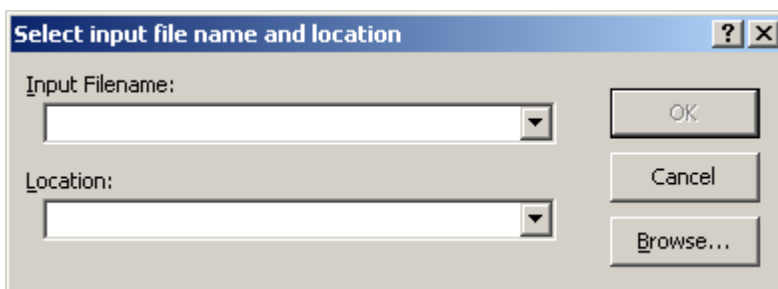
- 1 Select the **View Test Setup** button  from the toolbar.

The Test window appears.



- 2 Select the **Add Test Input** button.

The **Select input file name and location** dialog box appears.

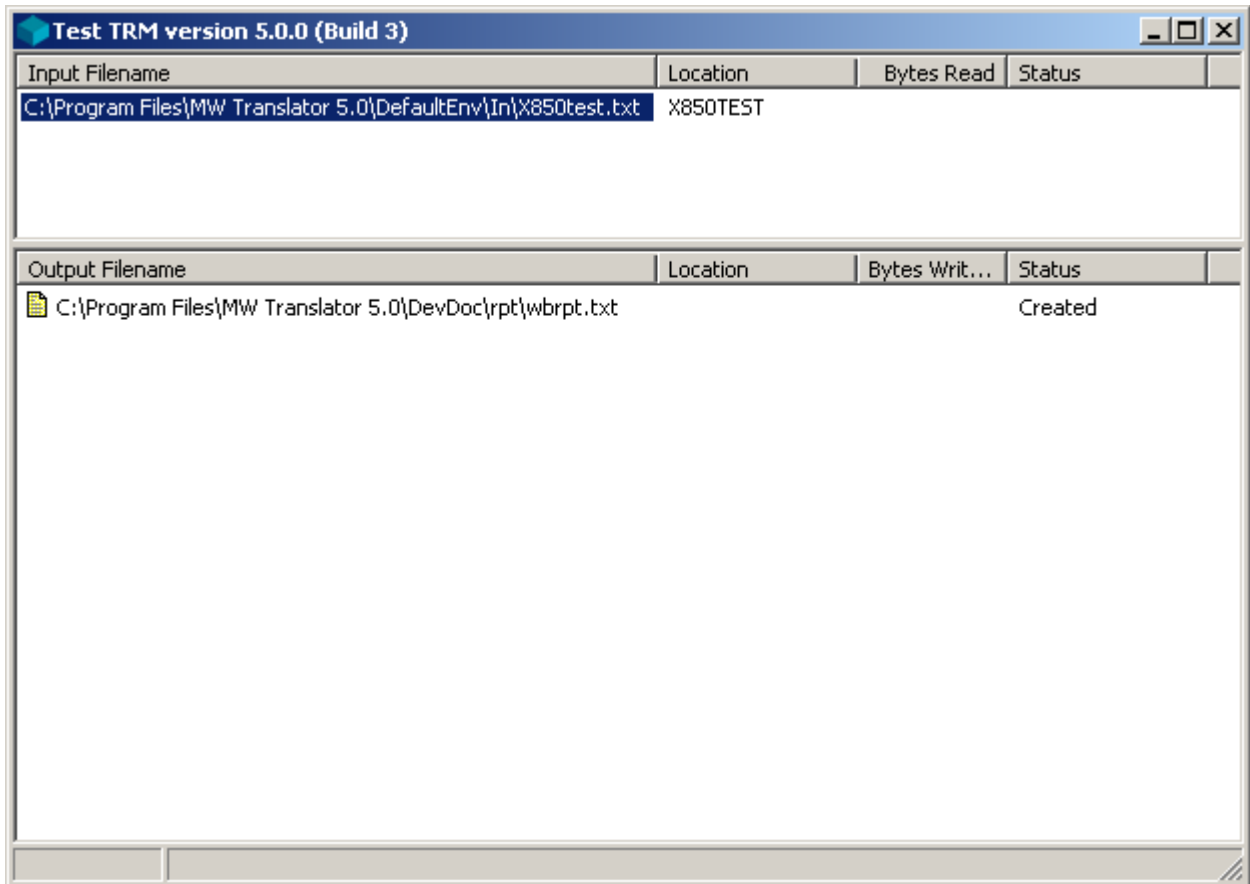


- 3 Use the **Browse** button to select an input file for the **Input Filename** dialog box.
- 4 Depending on where you have defined the input wrapper for standard identification, use the **Location** field as follows:
 - a) If you defined the input wrapper in the <Default> location, you can leave the field blank.




b) If you defined the input wrapper in a location other than <Default>, select the **Browse** button to locate it.

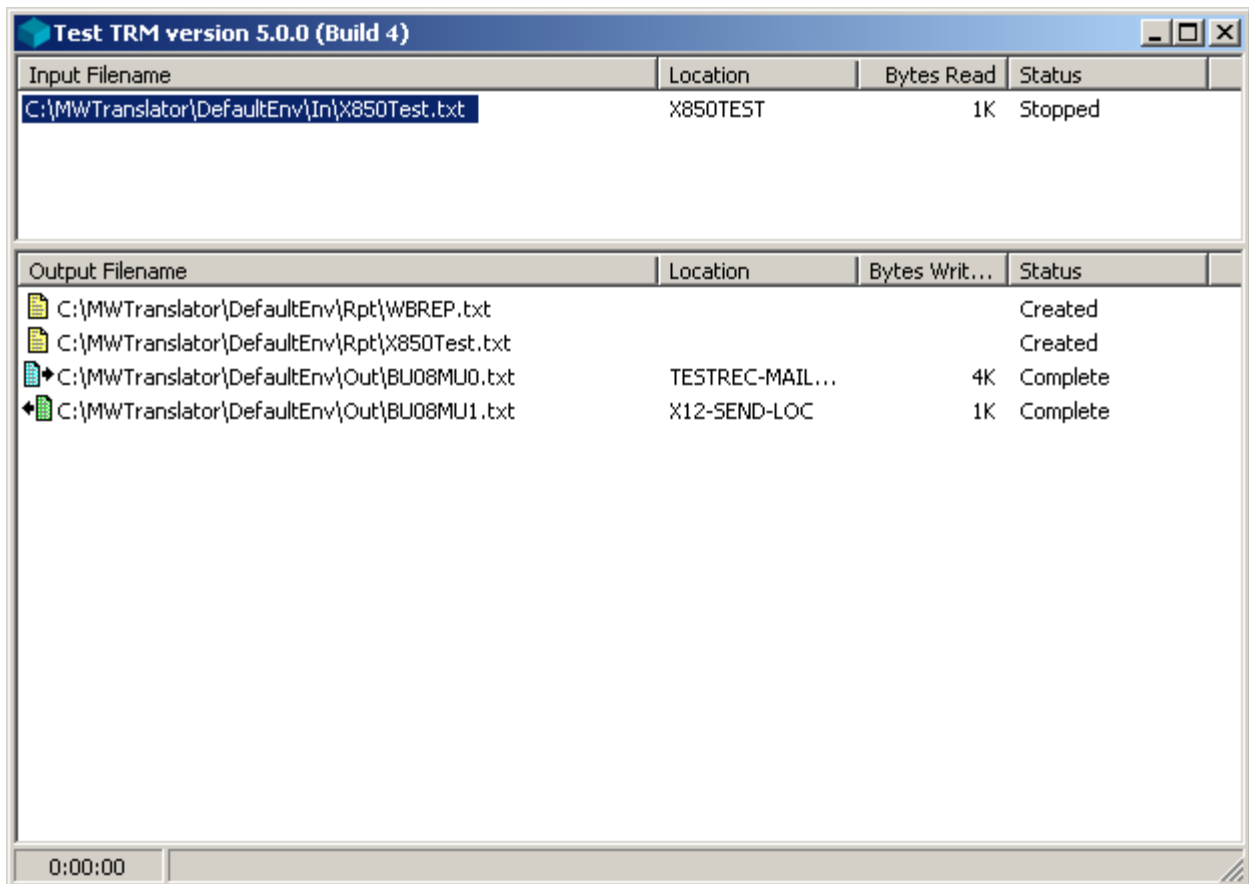
5 Select **OK**.

The Test window reappears with input file on the list.







6 Select the **Run Test** button .

The test results appear on the window. There will be 2 report files, identified by the report icon , and at least one output file, identified by the output file icon . If you are creating acknowledgments, you should also have a file containing the acknowledgment, identified by the acknowledgment icon . You can double-click any of these text files to display the contents in a word editor.



The screenshot shows a window titled "Test TRM version 5.0.0 (Build 4)". It contains two tables: "Input Filename" and "Output Filename".

Input Filename	Location	Bytes Read	Status
C:\MWTranslator\DefaultEnv\In\X850Test.txt	X850TEST	1K	Stopped

Output Filename	Location	Bytes Writ...	Status
 C:\MWTranslator\DefaultEnv\Rpt\WBREP.txt			Created
 C:\MWTranslator\DefaultEnv\Rpt\X850Test.txt			Created
 C:\MWTranslator\DefaultEnv\Out\BU08MU0.txt	TESTREC-MAIL...	4K	Complete
 C:\MWTranslator\DefaultEnv\Out\BU08MU1.txt	X12-SEND-LOC	1K	Complete

At the bottom left of the window, there is a timer showing "0:00:00".

During testing, you can queue up multiple files, and either run them in the order listed or run them individually. An input file can contain multiple interchanges. Depending on your configurations, you may have several output files, at least one for each interchange.

It may be that when you are ready to test your configurations, those responsible for supplying representative test data are not able to provide it for you. In such cases, you simply make up a test file based on what information you currently have. You can use this data to test your configurations. When the official test data is available, you will retest using it.

Begin simply and add complexity as you progress. When you are satisfied with your results, you will probably want to run tests on your target system, which is in an environment similar to the one you use for production. This is where you will do stress testing, for example.

For more information, refer to the section entitled, *Test Reference* (on page 761).

When you have run a test, you should review the processing report.

Reviewing Your Processing Report

Your processing report should contain all the information that you will need to determine the success of your test. If you are not getting enough information or you do not have a report at all, refer to the topic, *Setting Report Options* (on page 388).

NOTE: The file name of the processing report is that of the input file. Therefore, every time you run the same input file, the previous report is overlaid with the new report. If you want to save a particular version of a processing report, you should rename the file before you run another test using the same input file.

The processing report contains several sections. An example of each section appears with each following step. If you have requested to display file names on the report, these appear between the markers, <<< and >>>. The sample report here shows you the basic processes.

For more information about what the TRM is doing, refer to the topic, *How the TRM Processes Documents* (on page 46).

Header Information

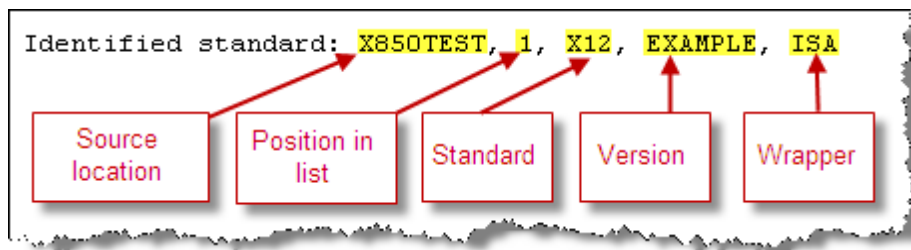
This section contains information about the start of the translation.

```
Test Runtime Module Version 5.0.0(Build 4)
Start of translation 2010/02/16 12:05:20
Source location: X850TEST Input name: C:\MWTranslator\DefaultEnv\In\X850Test.txt
```


Label	Description
Test Runtime Module Version	The version and build of the TRM. The build number may not be the same as the build number for the Workbench interface.
Start of translation	The date (year/month/day) and time (hour:minute:second) the translation began.
Source location:	The name of the source location that contains the wrapper definition used to parse the inbound wrapper.
Input name:	Location of the input data.

Identified Standard Information

This section shows which standard has been identified.



Label	Description
Identified standard	<p>The first element is the name of the source location.</p> <p>The second element is the numeric position in the list of wrappers for the location.</p> <p>The third element is the name of the standard to which the wrapper belongs.</p> <p>The fourth element is the name of the version in the standard.</p> <p>The last element is the name of the wrapper itself.</p>

For more information about this process, refer to the topic, *Step 1. Identify the Incoming Standard* (on page 47).

Processing Wrapper

This section displays information parsed from the various levels of wrappers.

```
>>> Wrapper set definition file name:
C:\MWTranslator\DevDoc\cfg\x12/W-X12-EXAMPLE-ISA.txt <<<
Processing wrapper, level: 1 Control Reference: 000010001
Sender ID: ICH-SEND-ID Qual: ZZ Key: 1243535703
Recipient ID: ICH-REC-ID Qual: ZZ Key: 1132317559

Processing wrapper, level: 2 Control Reference: 100010001
Partner Relationship Name: Example X12 Relationship
Sender Name: Example X12 Sender
ID: FG-SEND-ID Key: 2060521
Recipient Name: Example X12 Recipient
ID: FG-REC-ID Key: 6303532
```

Label	Description
Wrapper set definition file name	Name of the generated text file that contains the definitions the TRM uses to parse the inbound wrapper.
Processing wrapper, Level	The level of the wrapper being processed, which may contain references to partners and control numbers. Level 1 is the highest level, called the interchange level in X12 and EDIFACT standards, and level 2 is called the Functional Group level.
Control reference	The control number in the specified level of the header wrapper, if used.
Sender ID, Qual	When a partner ID is used for the sender, these are the values that were received.
Recipient ID, Qual	When a partner ID is used for the recipient, these are the values that were received.
Key	This is the reference number to identify this partner at this level in the partner text file.
Partner Relationship Name	When the partner IDs for both sender and recipient match a configured partner relationship, this is the name of the configuration.
Sender Name, ID	When the lowest level of wrapper has been parsed, and the IDs match a configured partner, this is the name of the configuration that is used for the sender.

Label	Description
Recipient Name, ID	When the lowest level of wrapper has been parsed, and the IDs match a configured partner, this is the name of the configuration that is used for the recipient.

For more information about this process, refer to the topics:

- *Step 3. Parse and Validate One Level of the Incoming Wrapper* (on page 50)
- *Step 4. Look for Defined Partner IDs for the Sender and Recipient at this Level* (on page 53)
- *Step 5. Find Partner Profiles for the Sender and Recipient Based on Partner IDs* (on page 55)
- *Step 6. Find a Partner Relationship Definition, If One Exists* (on page 57)
- *Step 7. Repeat Steps 3-6 for Remaining Wrapper Levels 2 through 4, if They Exist* (on page 57)

Trade Agreement Found

This section displays the name of the trade agreement found, the information used for matching, and where the trade agreement was found: on a partner relationship, recipient partner, or standard ID configuration. When a trade agreement is not found, the report specifies within which definitions the TRM searched for one.

```
>>> Document definition file name:
C:\MWTranslator\DefaultEnv\Cfg\x12/D-X12-EXAMPLE-850.txt <<<
Trade Agreement found on Partner Relationship: EXAMPLE-X850
Identified document: 850 Version: 003030 Agency: X
Control Reference: 0001
```

Label	Description
Document definition file name	Name of the generated text file that contains the definitions the TRM uses to parse the document.
Trade Agreement found on	Type of configuration on which the trade agreement was configured. The name of the Trade Agreement Profile follows the colon.
Identified document	The inbound data that matched the Doc ID on the Trade agreement Profile.
Version	The inbound data that matched the Version on the Trade Agreement Profile.
Agency	The inbound data that matched the Agency on the Trade Agreement Profile.
Control Reference	The control number in the document header, if used.

For more information about this process, refer to the topic, *Step 9. Find a Trade Agreement Profile* (on page 61).

Translation Output (Documents)

This section displays the name of the first output file, since there can be several, and the source and destination locations of that output. Note that the destination location listed in the report is the location that the TRM passes to a target messaging system used for production processing, such as MessageWay or Exchange. The messaging system will use this address to route the output.

```
>>> Wrapper set definition file name:
C:\MWTranslator\DefaultEnv\Cfg\appl\W-EXAMPLE-1-FWRAP.txt <<<
>>> Map file name: C:\MWTranslator\DefaultEnv\Cfg\map\M-EXAMPLE-FWRAP.txt <<<
>>> Document definition file name:
C:\MWTranslator\DefaultEnv\Cfg\appl\D-EXAMPLE-1-FPO.txt <<<
>>> Map file name: C:\MWTranslator\DefaultEnv\Cfg\map\M-EXAMPLE-X850.txt <<<
Translation Output stream created: C:\MWTranslator\DefaultEnv\Out\BU08MU0.txt
Output number: 1 Source: TESTSEND-MAILBOX Destination: TESTREC-MAILBOX

>>> Control reference file name: C:\MWTranslator\DefaultEnv\Cfg\CTRLREF.TXT <<<
Map Name: EXAMPLE-X850
```

Label	Description
Wrapper set definition file name	Name of the configuration text file that the TRM uses to create the outbound wrapper.
Map file name	Name of the map text file that the TRM uses to create the output wrapper.
Document definition file name	Name of the configuration text file that the TRM uses to create the outbound document.
Map file name	Name of the map text file that the TRM uses to create the output document.
Translation output stream created	File created that will contain the output data. Since multiple documents may be included in multiple wrappers within an interchange, the TRM determines when it must close this file and create new ones for remaining information.
Output number	This is the sequential number of the output file created from this input stream.
Source	This is the address that identifies the source of the output, which may be needed by the recipient or the messaging system that will send the data.
Destination	This is the address that identifies where the output will be sent.

Label	Description
Control reference file name	Name of the file used to generate control references, when necessary.
Map name	Name of the map in the configuration database used to create the output document. These are the definitions from which the Workbench generates the map text file.

For more information about options available to generate control references, refer to the *MW Translator Operator Guide and Reference*. For more information about this part of the translation process, refer to the following topics:

- **Step 10. Parse the Incoming Document(s) (Contents Validation and Translation)** (on page 68)
- **Step 14. Determine Routing and Miscellaneous Information (Translation)** (on page 91)

Status of Incoming Wrapper Level 3, Document

This section displays the document status, as well as the control reference from the wrapper trailer for this document. The status is the result of the compliance check, based on various criteria of the incoming standard.

```
Document Status: A Control Reference: 0001
```

Label	Description
Document Status	This is the status of the inbound document assigned by the TRM, where: <ul style="list-style-type: none"> ▪ A = accepted, with or without errors ▪ R = rejected, continue processing if possible or abort ▪ S = security reject, continue processing
Control Reference	This is the control number or reference for the document that was parsed from the inbound trailer wrapper.

Additional Output

The report includes processing information for any additional documents generated in this output stream, and will specify any additional output streams created. There is only one output file going to the recipient in this example, but it contains three documents. This is the information for the two additional documents. The TRM looks for a new trade agreement each time it encounters a new input document. In this test example, the three input documents happen to be the same, so the trade agreements it found are also the same.

If any of these documents needed to be routed to a different location than the previous document, a new output file would be created. For more information about when the TRM creates new output files, refer to the topic, *Step 17. Repeat steps 14 and 16 for each defined output (translation)* (on page 97).

```
Trade Agreement found on Partner Relationship: EXAMPLE-X850
Identified document: 850 Version: 003030 Agency: X
Control Reference: 0002

Map Name: EXAMPLE-X850

Document Status: A Control Reference: 0002

Trade Agreement found on Partner Relationship: EXAMPLE-X850
Identified document: 850 Version: 003030 Agency: X
Control Reference: 0003

Map Name: EXAMPLE-X850

Document Status: A Control Reference: 0003
```

For explanations of the information in this section of the report, refer to the following topics:

- *Trade Agreement Found* (on page 397)
- *Translation Output (Documents)* (on page 398)
- *Status of Incoming Wrapper Level 3, Document* (on page 399)

Status of Incoming Wrapper Level 2, Functional Group

The report displays the status of the inbound functional group level trailer wrapper, together with its control reference, if one exists. These statuses typically appear in acknowledgments for public standards, such as the 997 or CONTRL.

```
Wrapper level 2 Status: A Control Reference: 100010001
```

Label	Description
Wrapper level 2 Status	This is the status of the inbound functional group level wrapper, where: <ul style="list-style-type: none"> ▪ A = accepted, with or without errors ▪ R = rejected, continue processing if possible or abort ▪ S = security reject, continue processing
Control Reference	This is the control number or reference for the functional group level wrapper from the inbound trailer.

Translation Output (Acknowledgments)

If you generate acknowledgments, this section displays the file name of the output stream that would be created, and the source location and the destination location of the acknowledgment. Note that the destination location for the acknowledgment listed here is the location that the TRM passes to a target messaging system used for production processing, such as MessageWay or Exchange. The messaging system will use this address to route the acknowledgment.

```
>>> Wrapper set definition file name: C:\Program Files\MW Translator
5.0\DevDoc\cfg\x12/W-X12-EXAMPLE-ISA.txt <<<
>>> Map file name: C:\Program Files\MW Translator 5.0\DevDoc\cfg\map/M-EXAMPLE-ISA.txt <<<
>>> Document definition file name: C:\Program Files\MW Translator
5.0\DevDoc\cfg\x12/D-X12-EXAMPLE-997.txt <<<
>>> Map file name: C:\Program Files\MW Translator 5.0\DevDoc\cfg\map/M-EXAMPLE-X997.txt <<<
Acknowledgment Profile found on Partner Relationship: EXAMPLE-X997
Functional Acknowledgment Output stream created: C:\Program Files\MW Translator
5.0\DevDoc\out\ACIVIML.txt
Output number: 2 Source: X12-REC-LOC Destination: X12-SEND-LOC

>>> Control reference file name: C:\Program Files\MW Translator 5.0\DevDoc\cfg\ctrlref.txt <<<
```

Label	Description
Wrapper set definition file name	Name of the generated text file that the TRM uses to create the outbound wrapper for the acknowledgment.
Map file name	Name of the map text file that the TRM uses to create the output wrapper for the acknowledgment.
Document definition file name	Name of the generated text file that the TRM uses to create the acknowledgment.
Map file name	Name of the generated text file that the TRM uses to map the acknowledgment.

Label	Description
Acknowledgment profile found on	Where the acknowledgment was found, such as on a partner relationship configuration, and the name of the acknowledgment profile.
Functional acknowledgment output stream created	File created that will contain the output acknowledgment. Since this acknowledgment has a different destination than the other output documents, it is created as a separate output file.
Output number	This is the sequential number of the output file created from this input stream.
Source	This is the address that identifies the source of the acknowledgment.
Destination	This is the address that identifies where the acknowledgment will be sent.
Control reference file name	Name of the file used to generate control numbers, when necessary.

For more about this process, refer to the following topics:

- *Step 18. Create Backward Acknowledgments, If Required* (on page 99)
- *Step 19. Determine Routing for Backward Acknowledgments* (on page 103)

Status of Incoming Wrapper Level 1, Interchange

This section displays the status of the first, interchange level of the inbound wrapper. The TRM determines the status based on defined criteria for the standard.

```
Wrapper level 1 Status: A Control Reference: 000010001
```

Label	Description
Wrapper level 1 Status	This is the status of the inbound interchange wrapper, where: <ul style="list-style-type: none"> ▪ A = accepted, with or without errors ▪ R = rejected, continue processing if possible or abort ▪ S = security reject, continue processing
Control Reference	This is the control number or reference that was parsed from the interchange level wrapper of the inbound trailer.


End of Report

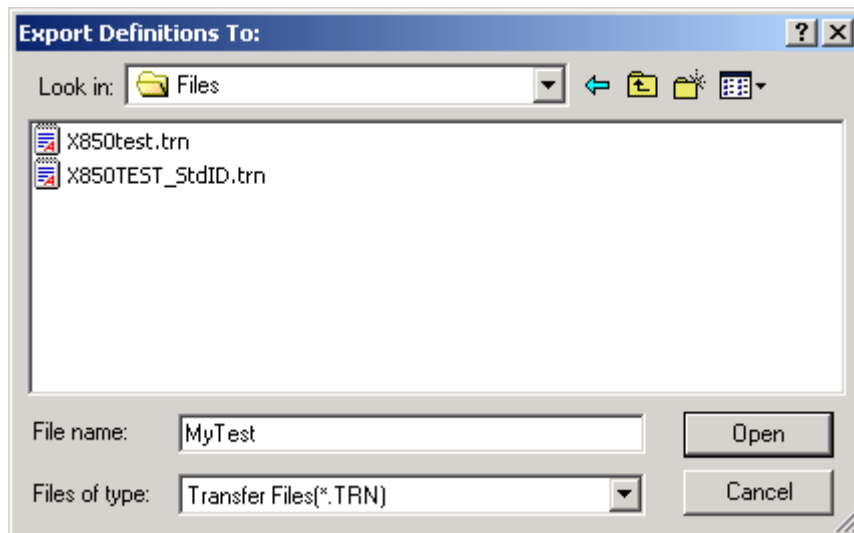
This is the date and time the translation finished. If you do not see this line, your report is incomplete.

```
>>>> End of Translation 2009/08/08 14:15:42 <<<<<
```

Exporting Definitions

The Workbench provides a way to save all definitions related to a particular test. This is helpful if you want to move definitions to another test database. You can also use the feature to provide customer support with files that they can use to help you resolve problems.

After you have run a test, you can select the **Export Translation Definitions** button  from the task bar. The **Export Definitions To** dialog box appears.



Select a location where you want to put the file, such as in the subdirectory **Files**, and enter the file name.

You can now use the **Import** command to add these definitions to another database, or you can add these definitions to your current database after renaming them.

This page intentionally blank

Administration

Overview of Administration

You have considerable flexibility to create and use various database environments. This helps to keep projects discrete and separate from one another, improving organization and reducing interference from other configurations. When you test or run production, the MW Translator Runtime Module uses a directory structure associated with an environment to find the information it needs for processing.

The environment concept organizes the information required to support a configuration and testing database. When you select the **Full MW Translator** option during the install process, you install the following components:

- A default database environment (DefaultEnv)
- Operator program
- Workbench program

Environments have subdirectories. These subdirectories and their contents are listed in the following table:

Subdirectory	Contents
CFG	All of the text files created from the database configurations that are used by the Translator Runtime Module (TRM) to process the data.
DB	Database files
FILES	Definitions that have been exported, which can then be imported
IN	Files used as input to TRM processing
OUT	Files created from TRM processing
RPT	Reports from TRM processing

You can then use the install program to provide additional environments by selecting the option, **New Database Environment**. This information describes the typical administration tasks:

- Creating environments
- Moving information to and from the databases
- Maintaining databases
- Transferring MW Translator configurations to MessageWay

NOTE: The database structure itself has changed for 4.0. In earlier versions there was a distinction between Workbench environments, available only to the Workbench, and Master environments, available to both the Workbench and the MW Translator Operator program. With MW Translator 4.0, there is only one type of database environment, accessible from both the Workbench and the MW Translator Operator program. When you import files or access databases from prior releases, The MW Translator Workbench converts them to 4.0 database environments. The Operator program does not have the ability to convert database environments.

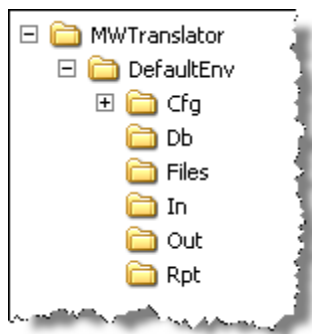
Creating and Using Various Environments

This is a very powerful feature but a simple concept. Use the installation program to create additional environments to separate your various projects. Although you can still create stand-alone databases by duplicating existing ones and then point to the new database using the Modify Options window, you must share the same configuration files among all databases. To avoid such confusion and potential conflict, entire environments are created when you use the install process or when you convert stand-alone databases.

Creating Environments

In Windows Explorer, you can see the default environment, DefaultEnv, that is created for each installation of the Workbench.

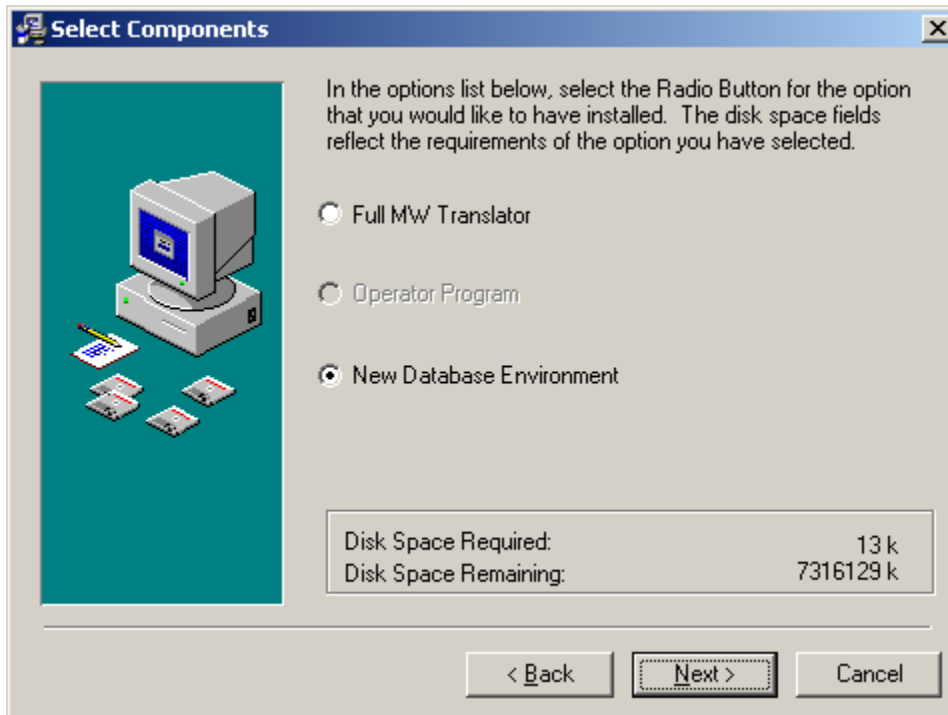
NOTE: As of release 5.0, the environments are created in the root directory, *not* in \Program Files.



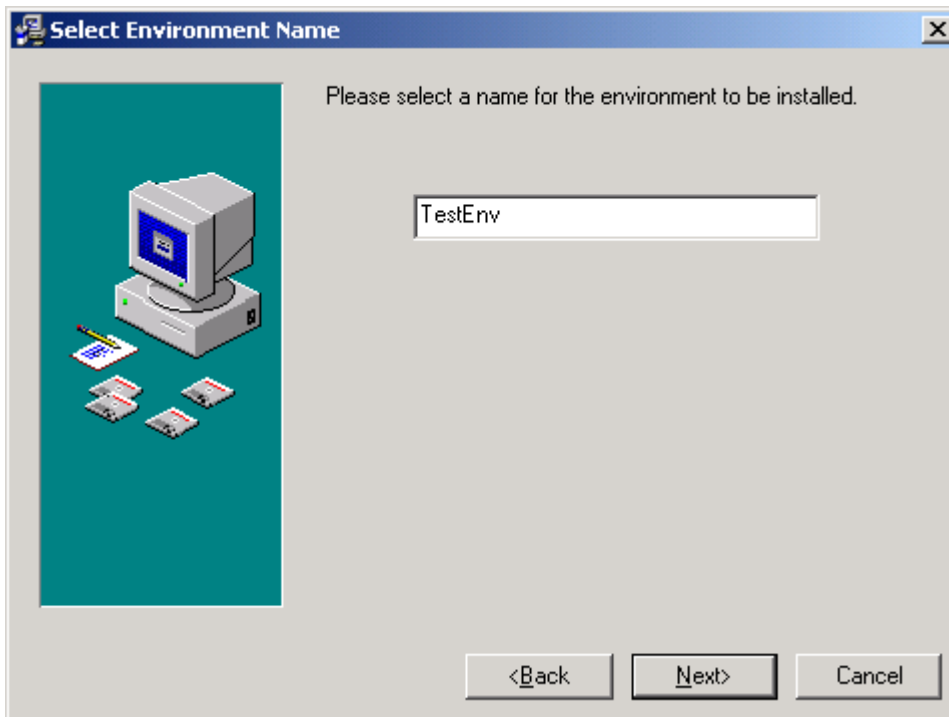
IMPORTANT: Stop the Workbench before you perform this task.

To create another environment:

- 1 Start the install program, and select **New Database Environment**.

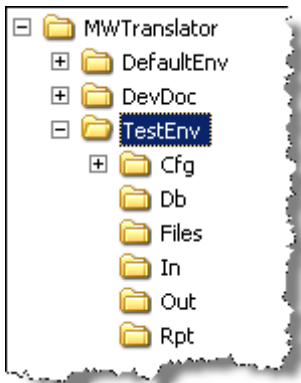


- 2 Type the name of your environment.

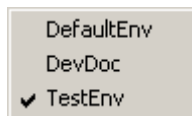


- 3 Click **Next**.

The environment is created with its appropriate directory structure.



This process also adds the new environment to the environment selection list. From the Workbench or




Operator Program, you use this drop-down list to change to a different environment.

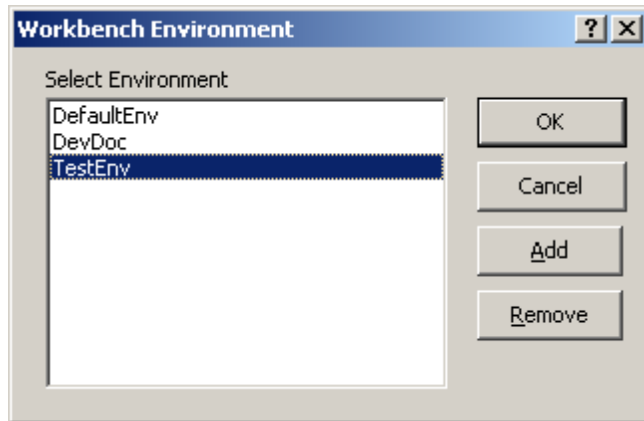
You can also choose to add an existing, stand-alone database to the list using **File>Select Environment**.

When you do so, if the database is stand-alone and not part of an environment, the Workbench creates an environment and moves the **DB** subdirectory beneath the environment directory.

Using Environments


When you install environments from the install program, the new environments automatically appear on the list. You maintain the list by selecting **Select Environment** from the **File** menu or the **Select**

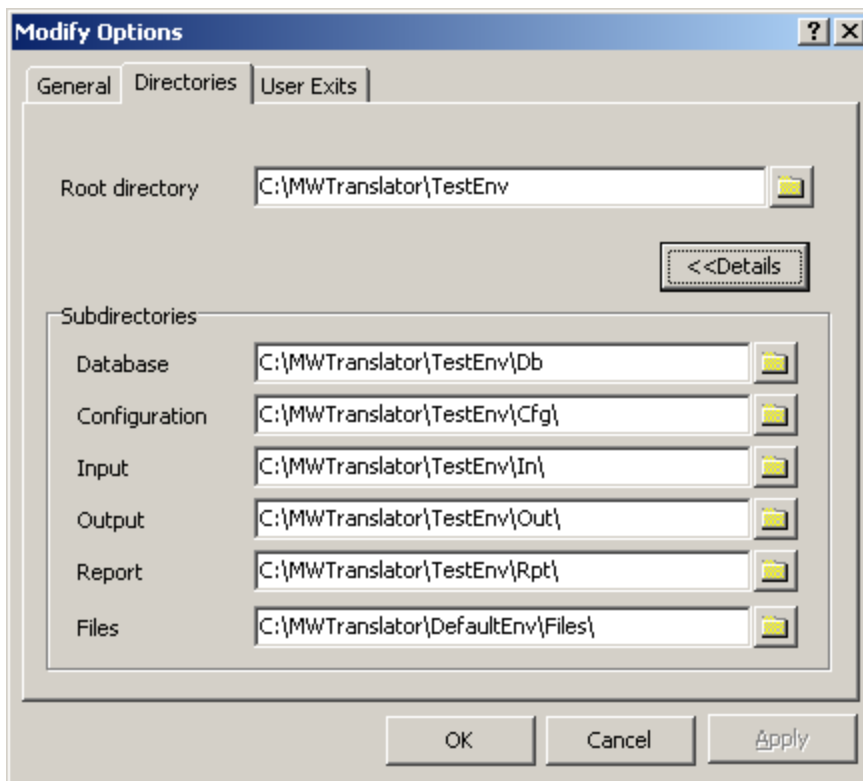
Environment button  from the toolbar. When the dialog box appears, you can add environments or remove environments from the list.



Removing environments from the list does not delete the files. If you want to delete the environment directory and files, you must do so manually from Windows Explorer. When you work with an environment, any configuration or testing input or output remains within the environment.

There is another feature in the Workbench that allows you to specify file locations: the **Directories** tab of the Modify Options window. You access this window by selecting **Options** from the **File** menu or the

Options button  from the toolbar. All subdirectories within an environment begin by pointing to the same directory path. You can change where the Workbench looks for this information, which you may want to do if someone moves a database that serves multiple users to a different location on the LAN.



Moving Environments

When you create an environment, it is always created as a subdirectory within the installation directory. To move an environment to a different location:

- 1 From the Workbench, **remove the environment name from the environment list** (on page 528).
- 2 From Windows Explorer:
 - a) Cut or copy the database environment from its original location.
 - b) Paste the environment in the new location.
- 3 From the Workbench, **add the environment name to the environment list** (on page 528).

IMPORTANT: If you move an environment before you remove it from the environment list, you may receive an error when the Workbench attempts to access the environment, because it is looking for it in the previous location. To fix the problem, you must *identify the environment's new location* (on page 519).

Deleting Environments

- 1 To delete MW Translator database environments, proceed as follows:
- 2 From the Workbench, *remove the environment name from the environment list* (on page 528).
- 3 From Windows Explorer:
 - a) Cut or copy the database environment from its original location.
 - b) Paste the environment in the new location.

Moving Information to and from Databases

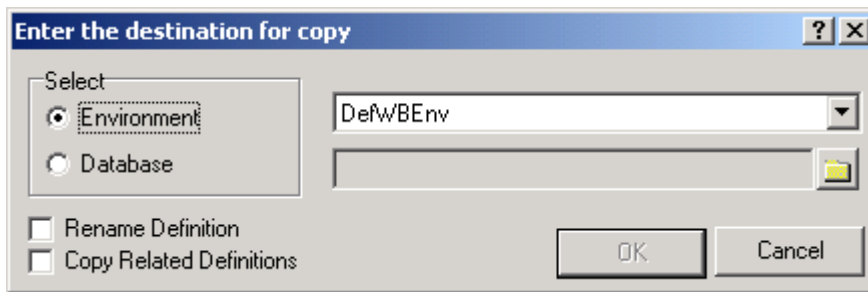
The default Workbench environment uses a database structure to store configurations. There are three utilities to move information to, from, and among databases:

- **Copy** moves information within the same database or from one database to another within the same Network Neighborhood, copying only selected files. For more information, refer to the topic, *Copy Command* (on page 492).
- **Export** moves configurations to a flat file, exporting all related files. For more information, refer to the topic, *Export Command* (on page 497).
- **Import** moves configurations from exported files into databases, importing selected files. For more information, refer to the topic, *Import Command* (on page 501).

Copying Configurations

You can copy configurations between databases in the same local network. This procedure is more efficient than moving the definitions in and out of flat files using import and export commands. When you copy definitions to the same database environment, you must rename them by selecting the **Rename Definition** box. For specific instructions, refer to the topic, *To Copy a Definition* (on page 496).

When the dialog box appears, you specify the target database or environment, and indicate whether or not you want to rename the definition or copy related definitions that are referenced by the main definition. The **Environment** field contains the environments appear on the **Select Environment** list. Selecting an environment is a bit easier than looking for the database file, but the result is the same.



Exporting Configurations

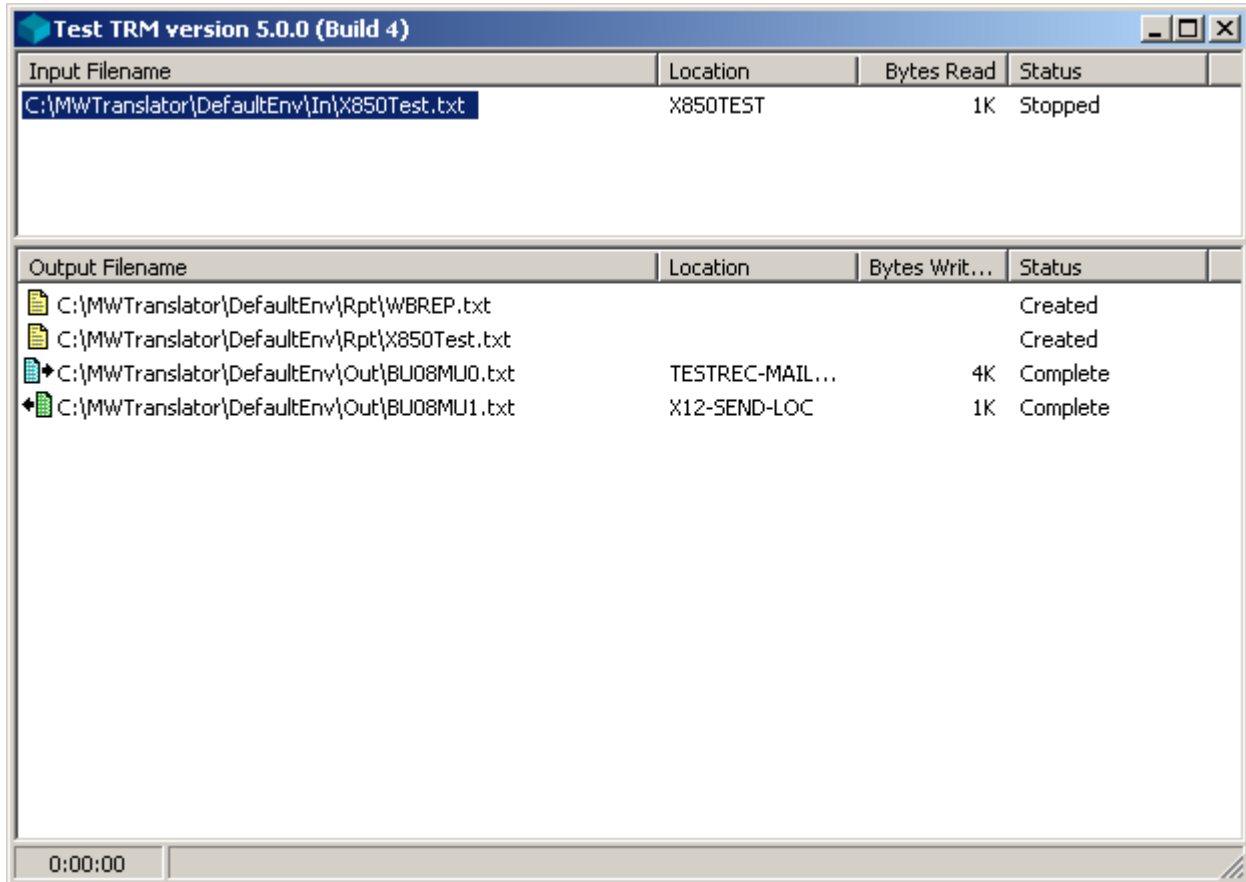
You can copy configurations to external files called TRN files using the **Export** command. This allows you to transfer definitions to an off-site database, such as to a customer support database. For specific instructions, refer to the topic, *To Export Definitions to Transfer Files* (on page 499).

There are two methods of export: one that you use from the Test window, and another that you use from the Workbench explorer windows. When you export configurations from the Test window after running a test, the utility copies all configurations required to run that test. When you export configurations from a Workbench explorer window, the utility copies only those configurations you have selected, and some related configurations as required. The following table shows the definitions that may be included in a transfer file using the **Export** command.

Export Entity	Definitions Included in TRN File
Test	All definitions required to run the test, including standard, standard version, wrappers, documents, segments, composite elements, elements, maps, trade agreement profiles, acknowledgment profiles, cross-references, locations/standard IDs, partners, partner relationships, and groups.
Document(s)	All definitions required for the document(s), including Standard, Standard Version, Document, Segments, Composite Elements, Elements

Export Entity	Definitions Included in TRN File
Wrapper(s)	All definitions required for the wrapper(s), including Standard, Standard Version, Document, Segments, Composite Elements, Elements
Map(s)	All definitions required for the map(s), including input and output documents and wrappers
Trade Agreement(s)	All definitions required for the trade agreement profile(s), including Trade Profile, map(s), document(s), and wrapper(s)
Acknowledgments	All definitions required for the acknowledgment profile(s), including Trade Profile, map(s), document(s), and wrapper(s)
Locations/StdID	Locations and standard ID
Partners	Partners
Partner Relationships	Partners and partner relationship(s)
Groups	Groups and partners

It is more efficient and less error-prone to export definitions after you run a test from the Test window than trying to figure out which files were used during the test, and copying or exporting these one by one.



The screenshot shows a window titled "Test TRM version 5.0.0 (Build 4)". It contains two tables. The first table, "Input Filename", has one row with the following data:

Input Filename	Location	Bytes Read	Status
C:\MWTranslator\DefaultEnv\In\X850Test.txt	X850TEST	1K	Stopped

The second table, "Output Filename", has four rows with the following data:

Output Filename	Location	Bytes Writ...	Status
C:\MWTranslator\DefaultEnv\Rpt\WBREP.txt			Created
C:\MWTranslator\DefaultEnv\Rpt\X850Test.txt			Created
C:\MWTranslator\DefaultEnv\Out\BU08MU0.txt	TESTREC-MAIL...	4K	Complete
C:\MWTranslator\DefaultEnv\Out\BU08MU1.txt	X12-SEND-LOC	1K	Complete

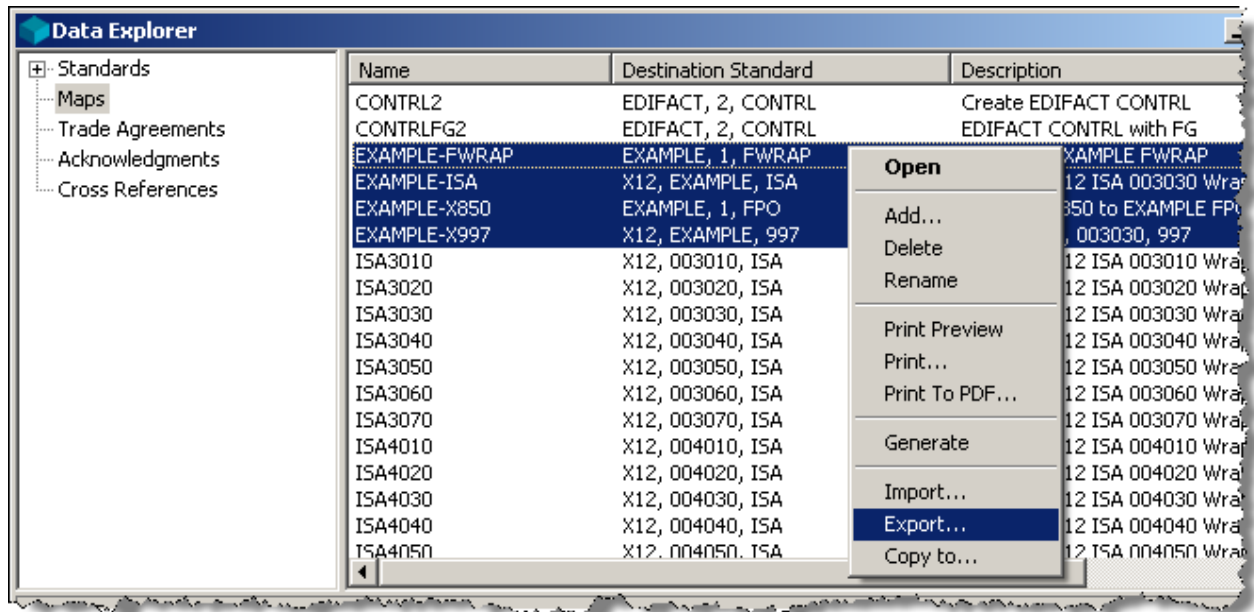
At the bottom left of the window, there is a timer showing "0:00:00".

After you run the test, you select **Export** from the **File** menu, or the **Export Translation Definitions** button

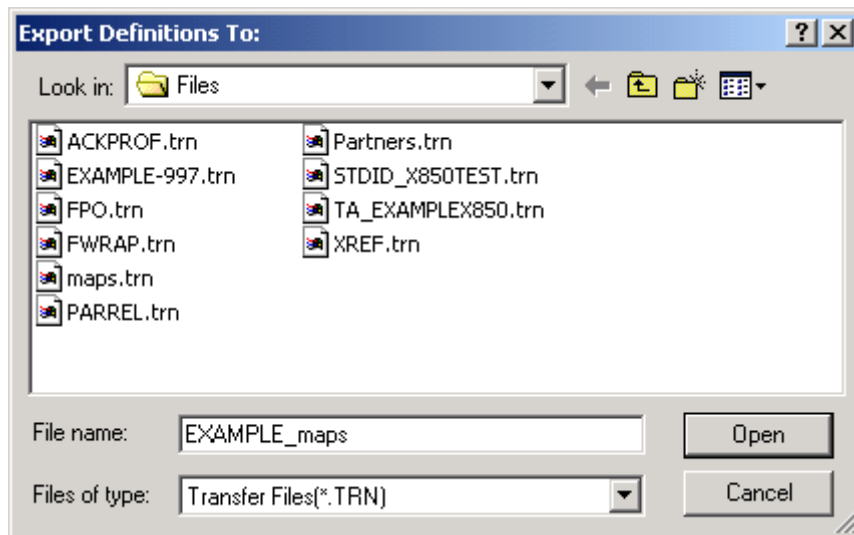


from the toolbar.

To export individual definitions from Data Explorer or Partner Explorer, select the definition(s) listed in the window and then select **Export** from the **File** menu, or right-click and select **Export** from the menu.



When the **Export Definitions To** dialog box appears, specify the file name for the TRN file to which the definitions are to be saved.



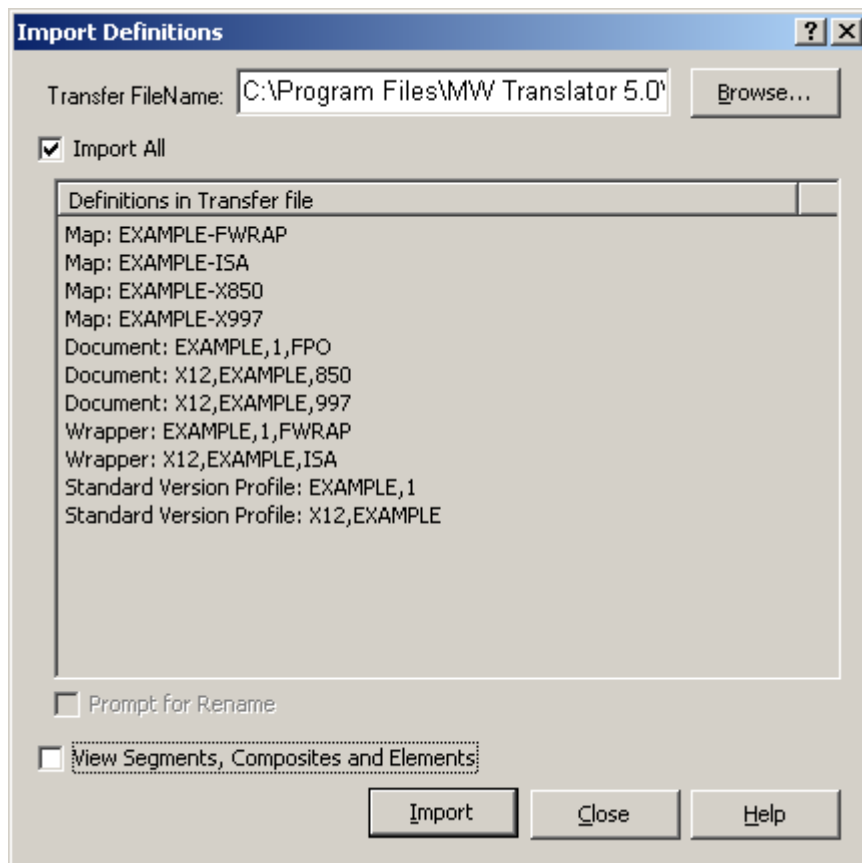
You can then import this file to another MW Translator Workbench system.

Importing Configurations

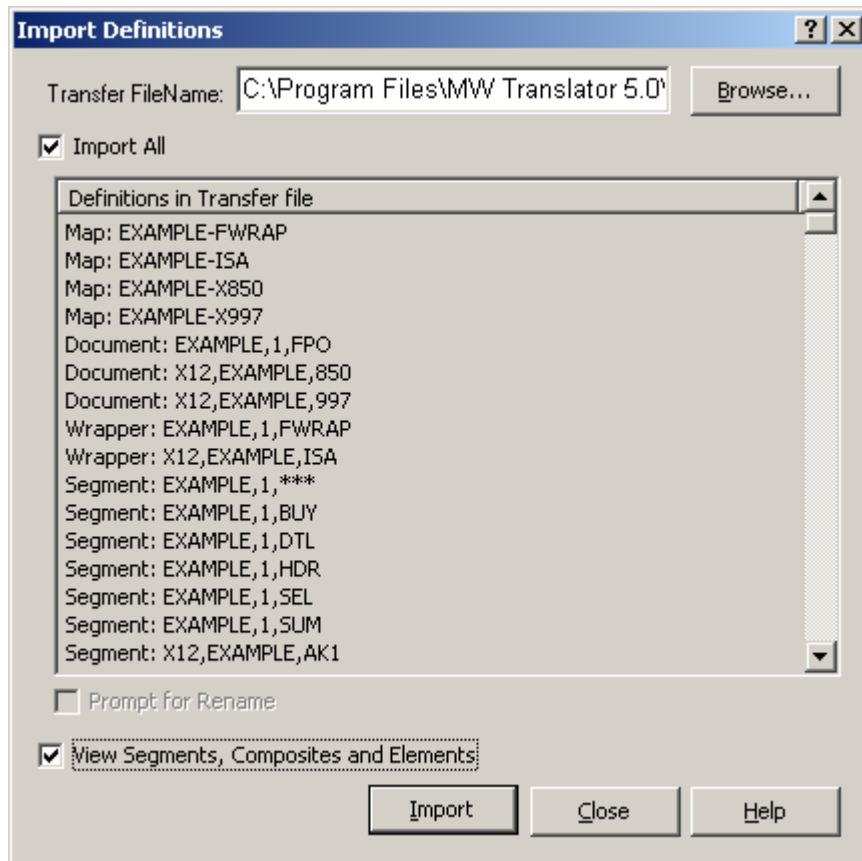
You may transfer configurations to environment databases using the **Import** command. You can import configurations from flat files that were created from a database using the **Export** command or from an XML DTD using the conversion routine, XML2DTD. For more information about the XML2DTD program, refer to the topic *Defining a Document from an XML DTD* (on page 369).

For specific instructions, refer to the topic, *To Import a Definition to a Database* (on page 505).

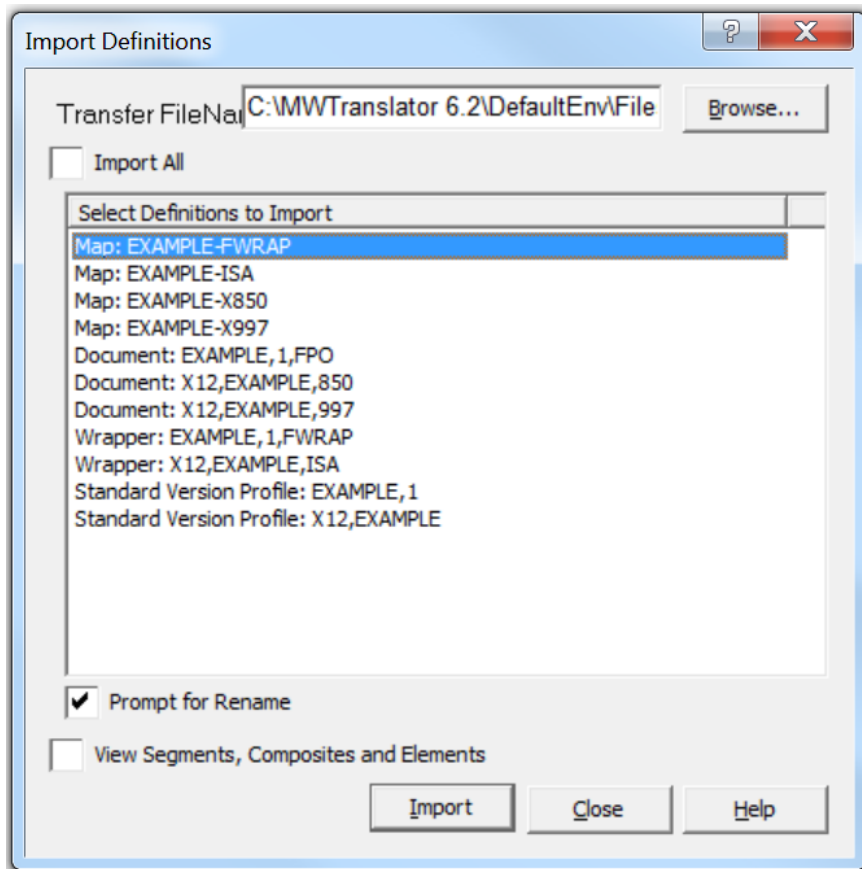
Initially when the **Import Definitions** dialog box appears, all files on the list are available for import. You only need to select the **Import** button to import all definitions. Notice that in the following example, the list includes not only the maps that were selected with the **Export** command, but also the documents and wrapper that the maps use.



You can also select definitions at a lower level of detail. To display more detailed definitions, select the **View Segments, Composites and Elements** box.

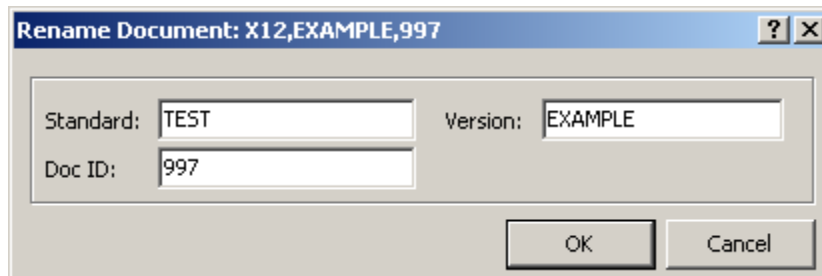


To import selected definitions, uncheck the **Import All** box, and select the ones required. Use the **SHIFT** key to select multiple contiguous definitions. Use the **CTRL** key to select discontinuous definitions. Select the **Prompt for Rename** box to change the name of the first definition displayed. Check boxes if you want to import them to a different standard and version, and click **Import**.

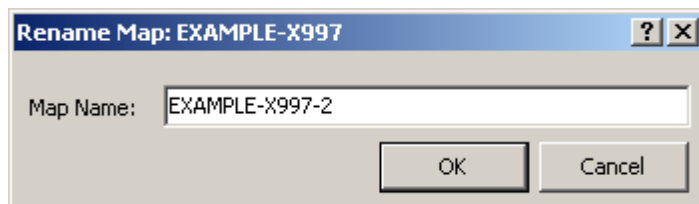


NOTE: Only the first definition within the Import Definitions window can be renamed. If you need to rename a definition that is not the first one, you will need to export only the definition you want to rename and import it using the rename option.

When the **Rename Wrapper** dialog box appears, enter the information requested, and click **OK**.



The **Rename Document: X12,EXAMPLE,997** dialog box contains three input fields: **Standard:** TEST, **Version:** EXAMPLE, and **Doc ID:** 997. At the bottom right are **OK** and **Cancel** buttons.



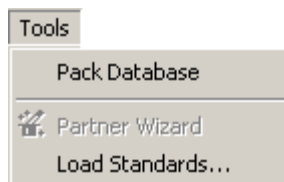
The **Rename Map: EXAMPLE-X997** dialog box contains one input field: **Map Name:** EXAMPLE-X997-2. At the bottom right are **OK** and **Cancel** buttons.

Use the Data Explorer or Partner Explorer windows to access the definitions you import.

IMPORTANT: Database versions are upwardly compatible but not downwardly compatible. You may import files from earlier versions of MW Translator to MW Translator 6.2 but you may not import files from MW Translator 6.2 to earlier versions of MW Translator database environments.

Maintaining Databases

If you are actively adding and deleting configurations from your database you can use the utility option under the **Tools** menu, **Pack Database**.



This releases physical space to the operating system for logically deleted information.

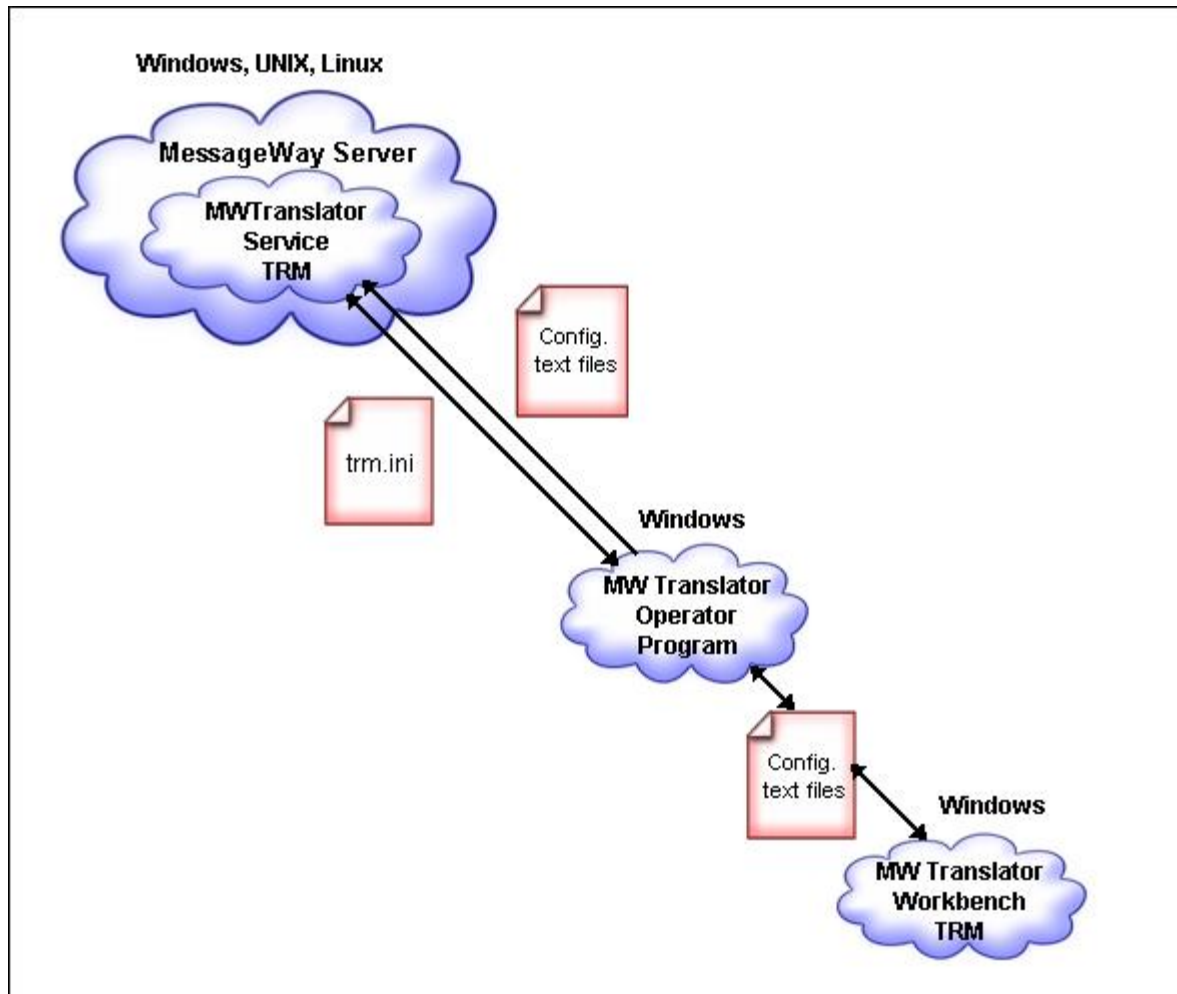
IMPORTANT: Make sure that you back up your databases. You can do this quickly by zipping them all to a file or by copying the entire DB subdirectory to another location.

Transferring MWTranslator Configuration Information to MessageWay

Once you have fully tested your configurations on the MW Translator Workbench, you will transfer the configuration text files to MessageWay using the Operator Program. The text files are stored in `../MessageWay/server/MWTranslator/cfg`.

For more information, refer to the MW Translator Operator Program online help or the *MW Translator Operator Program User's Guide and Reference*. The information transferred from the Operator Program to the Translator Runtime Module (TRM) running under MessageWay includes:

- Configuration text files created when you issue the **Generate** command in the Workbench or the Operator Program
- Changes to the TRM configuration file, `trm.ini`



This page intentionally blank

Troubleshooting

Runtime Exceptions

Three types of exceptions or errors occur during runtime: system errors, predefined compliance errors, and user-defined compliance or validation errors. System errors are generally numbered 9000 to 20000, except for some intervening security errors. Errors numbered 20001 and above are for user-defined validation errors.

System errors, by default, cause aborts. Some system errors, 9110, 9040, and 15001, cause a security reject of an interchange, which allows processing to continue in case there are multiple interchanges in the input stream.

Predefined compliance errors are numbered 1001 to 8999. The action performed depends on the setting in the Trade Agreement Profile for the current document, which can be either accept or reject.

User-defined errors can be invoked during parsing, which we refer to as user validation routines, or during a user exit. To invoke validation errors during parsing, you have two Edibasic methods: **Method Validate** and **Method ValidateAll**. Within the method, you include the error action using the **Edibasic Exception** function. User-defined errors can also be invoked during mapping, using the **Edibasic Exception** function. The **Edibasic Exception** function allows you to set the action for a particular error to accept, reject, or abort. If you do not define an action for user-defined errors, the TRM will use the default action of **Reject**. You can also use the **Edibasic Exception** function to invoke a particular system or predefined compliance error, and select the resulting action. Doing so does not override any action the TRM will take for system or predefined compliance errors that it generates during processing. Any undefined exceptions default to an action of abort.

The following table shows the type of error and its possible actions, some of which are used as defaults, and how the action is determined.

Error Type (range)	Action	Determined by
System (9000 - 20000)	abort (default)	TRM
9024	warning	TRM

Error Type (range)	Action	Determined by
9110, 9140, 9150, 15001	security	TRM allows individual interchanges within an input stream to be rejected with a security violation, and continue processing the other interchanges.
Compliance or configuration (1001-8999): <ul style="list-style-type: none"> ▪ 1001-5999, 7000-8999 Parse errors ▪ 6000-6999 Generate errors 	accept	Error Action setting on Trade Agreement Profile, Options tab.
	reject	Error Action setting on Trade Agreement Profile, Options tab.
User-defined (20001-30000)	accept	Edibasic Exception function.
	reject (default)	Edibasic Exception function. May be used as a default if there is no Exception function or no action specified for the function.
	abort	Edibasic Exception function.
Undefined	abort	TRM

Aborts may occur as a result of one of these types of events:

- Corrupt or missing configurations. May be caused by failed generate in the Workbench or by system errors.
- Control reference and logging database errors. Usually the result of improper configuration.
- Unable to identify standard. This may be avoided by configuring a catchall standard to handle junk.
- Unable to find a trade agreement. As above this may be avoided by creating a dummy trade agreement to handle documents that do not match any other trade agreement.
- User invoked abort. These are caused by Edibasic calls to Invoke() or user exit
- Calls to TRMInvoke() explicitly setting the action to ABORT.
- Missing or invalid user exits.
- System errors such as insufficient memory, I/O errors.
- Generation of non-compliant acknowledgment. (Generation of non-compliant output should result in the reject of the associated input document, functional group or interchange. The least amount possible will be rejected.)

In addition, the TRM must abort processing if it is incapable of properly handling any other behavior (for instance reject). This situation may occur if output user exits are used that don't support rollback and an error occurs during generation.

The results of each of the three types of actions is given in the following table:

Action	Result
accept	Accept with error. No action other than to report the error on the runtime report and possibly on an acknowledgment.
reject (default)	Reject the document or interchange. Whenever possible only the current document is rejected; however, in some cases (error in wrapper, error during mapping), the entire interchange must be rejected.
abort	Abort the processing of the current message. All output files already generated for that message are discarded.
warning	Note a change in the map that could cause a problem with processing. This does not stop processing, but you may not get the results you expected.

Translator Runtime Module (TRM) Exceptions

Exceptions generated by the Translator Runtime Module might occur when you are testing using the Workbench or running production on a target platform. These exceptions generate messages that will appear on the processing report.

Following is a list of predefined errors you might encounter during testing or production. This list does not include any user-defined exceptions you may have created. Each error is identified by a number, the possible action(s) that the TRM can take, the error text message, and a brief explanation, if needed.

NOTE: User-defined errors are number 20001 through 30000. You must provide your own documentation for such errors.

1001 Too few segment or loop repeats

Text	Too few segment or loop repeats
Error Action	Accept or reject
Error Type	Compliance
Processing Phase	Parse
Description	This exception would never occur. You would instead get error 1002 (Mandatory element missing). Users can enter a negative number of repeats, which should not be possible, and then you might get it. Otherwise, the minimum is zero or greater than zero, which is a result of the requirement designator (req), which is not zero when req is M (mandatory) or zero when req is O (optional) or C (conditional).

1002 Too many segment repeats

Text	Too many segment repeats Includes pertinent segment information, such as, Segment: LIN(47) Segment count: 839 Occurrences: 1000 Maximum allowed: 999
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse
Description	The segment has more than the maximum number of occurrences defined for the standard.

1003 Mandatory entity missing

Text	Mandatory loop/segment/element/composite missing Includes pertinent information, such as, Segment: LIN(47) Segment count: 32 Composite Seq: 6 Composite ID: C186 Element Seq: 2 Element ID: 6060
Error Action	Accept or Reject
Error Type	Compliance error.
Processing Phase	Parse
Description	Missing mandatory loop, segment, element, or composite.

1004 Segment contains too many elements

Text	Segment contains too many elements Includes pertinent information, such as, Segment: NAD(7) Segment count: 4
Error Action	Accept or Reject
Error Type	Compliance error
Processing Phase	Parse
Description	Segment contains more elements than are defined in the standard.

1005 Too many loop repeats

Text	Too many loop repeats Includes pertinent information, such as, Loop: LIN{47} Segment Count: 837 Occurrences: 1000 Maximum allowed: 999
Error Action	Accept or Reject
Error Type	Compliance error
Processing Phase	Parse
Description	The loop has more than the defined maximum occurrences.

1006 Too many composite repeats

Text	Too many composite repeats Includes pertinent information, such as, Segment: COM{14} Segment Count: 10 Occurrences: 10 Maximum allowed: 9
Error Action	Accept or Reject
Error Type	Compliance error
Processing Phase	Parse
Description	The composite element has more than the defined maximum occurrences.

1007 Too many element repeats

Text	Too many element repeats Includes pertinent information, such as, Segment: COB{52} Segment Count: 62 Occurrences: 10 Maximum allowed: 9
Error Action	Accept or Reject
Error Type	Compliance error
Processing Phase	Parse
Description	The element has more than the defined maximum occurrences. Repeating elements only appear in delimited standards.

1101 Element contains invalid character

Text	Element contains invalid character: #0D# Type: NO Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "234#0D##0A#GE"
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse
Description	Element contains character that is not permitted for that data type for the standard. This can happen if a delimited standard is not recognized as such, or if the delimiters received are incorrect.

1102 Data element is too long

Text	Data element is too long Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "374382938473823423"
Error Action	Accept or Reject
Error Type	Compliance error.
Processing Phase	Parse
Description	Number of characters in the element is greater than what is allowed by the element definition.

1103 Data element is too short

Text	Data element is too short Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "3"
Error Action	Reject
Error Type	Compliance error.
Processing Phase	Parse
Description	Number of characters in the element is less than what is allowed by the element definition.

1104 Data element contains invalid code value

Text	Data element contains invalid code value Includes pertinent information, such as, for Part 1 (if element uses partitions, such as X12 element 103) Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "3", or Element's Part 1 value: AAA (if element uses partitions, such as X12 element 103)
Error Action	Reject
Error Type	Compliance error.

Processing Phase	Parse
Description	You have chosen to validate element ID codes, and the referenced code does not match any of those listed for the specified element or for the specified partition of the element.

1105 Invalid date

Text	Invalid date
	Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "3"
Error Action	Reject
Error Type	Compliance error.
Processing Phase	Parse
Description	The date value is invalid for the specified format.

1106 Invalid time

Text	Invalid time
	Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "3"
Error Action	Reject
Error Type	Compliance error.
Processing Phase	Parse
Description	The time value is invalid for the specified format.

1107 Invalid value

Text	Invalid value Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "3"
Error Action	Reject
Error Type	Compliance error.
Processing Phase	Parse
Description	The value is invalid for the element.

1108 Required conditional element missing

Text	Required conditional element missing Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "3"
Error Action	Reject
Error Type	Compliance error.
Processing Phase	Parse
Description	The element is required by a condition, but it contains no value.

1109 Invalid delimiter found

Text	Invalid delimiter found while parsing element Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Delimiter: + Valid delimiters: *~
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse

Description Invalid delimiter found in element; probably a composite delimiter (and not in a composite) or a segment tag separator (if different from element separator).

1110 Unrecognized element tag

Text Unrecognized element tag
Includes pertinent information, such as,
Unrecognized element tag: 999
Current pos: 5
Segment: Segment count:

Error Action Accept or Reject

Error Type Compliance

Processing Phase Parse

Description The input value determined to be the element tag does not match the configuration definition.

1111 Component separator found in simple element

Text Component separator found in simple element
Includes pertinent information, such as,
Segment: UNG(3) Segment count: 5
Element Seq: 5 Element ID: 0048
Element value: "19:45"

Error Action Reject

Error Type Compliance

Processing Phase Parse

Description The component separator character was present in an element that is not a composite. For example, if you use the colon as a component element separator, and a colon appears in the data, as in this example of a time element, which is not a composite, you will receive this error.

1201 Composite conditional failed

Text	Composite conditional failed Includes pertinent information, such as, Exception # 1201 (Reject) - Composite conditional failed Segment: HI(56) Segment count: 50 Composite Seq: 1 Composite ID: C022 Cond: 1 P 03 04
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse
Description	The identified composite element has a condition of a specific type, in this case, paired (P), which means when one of the elements identified in the condition is present, all must be present. The condition number (1 P 03 04) is the number used by the standard to identify this condition. For X12, this notation means the first condition (1) on this composite is a paired condition (P) and both component elements 3 (03) and 4 (04) must be present.

1202 Composite conditional failed

Text	Composite conditional failed Includes pertinent information, such as, Exception # 1202 (Reject) - Composite conditional failed Segment: HI(56) Segment count: 50 Composite Seq: 1 Composite ID: C022 Cond: 2 E 08 09
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse
Description	The identified composite element has a condition of a specific type, in this case, paired (E), which means when one of the elements identified in the condition is present, all must be present. The condition number (2 E 08 09) is the number used by the standard to identify this condition. For X12, this notation means the second condition (2) on this composite is an exclusion condition (E) where elements 8 (08) and 9 (09) are mutually exclusive. When 8 is present, element 9 must not be, and when 9 is present, 8 must not be present.

1301 Segment conditional failed

Text	Segment conditional failed Includes pertinent information, such as, Segment: MEA(21) Segment count: 3 Cond: 2 C 03 04
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse
Description	The identified segment has a condition of a specific type, in this case, required (R), which means that at least one of the elements identified in the condition must be present. The condition number (2 C 03 04) is the number used by the standard to identify this condition. For X12, this notation means that if element 3 is present, then element 4 must also be present.

1302 Unrecognized segment

Text	Unrecognized segment Includes pertinent information, such as, Segment Data: AA*001*1000~ Current pos: Segment DTM(24) Segment count 10
Error Action	Accept or Reject
Error Type	Compliance error
Processing Phase	Parse
Description	This segment is not defined for this document. For fixed-length data, this happens when the maximum lengths of the elements do not match the length of the fields in the input data. This also happens when you do not account for CR/LF or NL characters. To tell MW Translator to ignore such characters between segments, refer to the topic, To Ignore Specific Characters between Segments During Parsing (on page 652).

1303 Segment out of order

Text	Segment out of order Includes pertinent information, such as, Segment Data: BGM*23432*4932~ Current pos: Segment: DTM(24) Segment count: 10
Error Action	Accept or Reject

Error Type	Compliance error
Processing Phase	Parse
Description	Segment is defined in the document, but the current position is beyond where the segment is allowed.

1304 Data Segment is too long

Text	Data Segment is too long Includes pertinent information, such as, Message: "JOBSEA RAYLOC CORR TN 2001-01-08CORES RETURN REQUEST CORR TN CRH811706CO00113MIDLAND MBI #108 20010 XXXXXX"
Error Action	Reject
Error Type	Compliance error
Processing Phase	Parse
Description	The parsing schema for the fixed-length standard has found that the segment exceeds the length specified based on the combined length of the elements.

1305 Segment conditional failed

Text	Segment conditional failed Includes pertinent information, such as, Segment: MEA(21) Segment count: 3 Cond: 6 E 08 03
Error Action	Accept or Reject
Error Type	Compliance error
Processing Phase	Parse
Description	The identified segment has a condition of a specific type, in this case, Exclusion (E), which means that at least one of the elements identified in the condition must be present. The condition number (6 E 08 03) is the number used by the standard to identify this condition. For X12, this notation means the sixth condition (6) on this segment is an exclusion condition (E) where elements 3 and 8 are mutually exclusive. If 3 is present, 8 should not be, or if 8 is present, 3 should not be.

1401 Invalid Bounded Loop Trailer Segment

Text	Invalid Bounded Loop Trailer Segment Includes pertinent information, such as, Loop Id Expected: 0100 Loop Id Found: 0200
Error Action	Accept or Reject
Error Type	Compliance error.
Processing Phase	Parse
Description	Found bounded loop trailer, but the loop ID does not match the ID in the loop header.

1402 Missing Bounded Loop Trailer Segment

Text	Missing Bounded Loop Trailer Segment Includes pertinent information, such as, Loop Id Expected: 0100
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse
Description	Bounded loop trailer was not found.

1901 A warning was reported during XML parsing of the input

Text	A warning was reported during XML parsing of the input.
Error Action	Accept
Error Type	Compliance
Processing Phase	Parse
Description	This error is often accompanied by others that cause a different action, such as an abort or reject error.

1902 An error was reported during XML parsing of the input

Text	<p>An error was reported during XML parsing of the input</p> <p>Includes pertinent information, such as,</p> <p>Exception # 1902 (Reject) - An error was reported during XML parsing of the input:</p> <p>line:7 col:35 Unknown element 'purchaseOrder2</p> <p>– or –</p> <p>line:21 col:13 Unknown element 'shipTo</p> <p>– or –</p> <p>line:35 col:14 Not enough elements to match content model : '((name,street,city,state),zip</p>
Error Action	Reject
Error Type	Compliance
Processing Phase	Parse
Description	<p>This error affects the interchange. It could be caused by a malformed tag, a missing element, or in the first case, when the MW Translator loop ID does not match the segment ID of the first segment in the loop. It will cause further processing to fail, with additional errors, such as 2011 (on page 442) "Invalid wrapper segment" and 7001 (on page 456) "Unable to find segments in the document."</p>

1903 A fatal error was reported during XML parsing of the input

Text	<p>A fatal error was reported during XML parsing of the input</p> <p>Includes pertinent information, such as,</p> <p>Exception # 1903 (Abort) - A fatal error was reported during XML parsing of the input</p> <p>line:1 col:20 Expected whitespace</p> <p>– or –</p> <p>line:1 col:6 No processing instruction starts with 'xml'</p> <p>– or –</p> <p>line:1 col:14 Expected a 'version=', 'encoding=', or 'standalone='</p>
Error Action	Abort
Error Type	Compliance
Processing Phase	Parse

Description Could be caused by malformed XML declaration (missing ? or uppercase characters).

1904 Skipping undefined XML element

Text Skipping undefined XML element
Includes pertinent information, such as,
Exception # 1904 (Warning) - Skipping undefined XML element - shipDate - within purchaseOrder

Error Action Accept

Error Type Compliance

Processing Phase Parse

Description Element received in input data is not defined in the MW Translator document, which may mean that a namespace or a namespace prefix is invalid

1905 Skipping undefined XML attribute

Text Skipping undefined XML attribute
Includes pertinent information, such as,
Exception # 1905 (Warning) - Skipping undefined XML attribute - orderDate - within purchaseOrder

Error Action Accept

Error Type Compliance

Processing Phase Parse

Description Attribute received in input data is not defined in the MW Translator document. Define the attribute in MW Translator within a segment as an element with a Category of **XML Attribut** on the Element Properties window.

When the attribute is a namespace, specify the namespace on the **XML** tab of the Segment Properties window with the syntax:
`xmlns[:prefix]="URI"`

This namespace must be defined on a segment before when it appears in the document, so it is typically defined on the first segment of the document, which represents the XML root element.

2001 Control reference mismatch

Text	Control reference on trailer does not match header Includes pertinent information, such as, Header control reference: 1000001 Trailer control reference: 1000002 Wrapper Name: ISA
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse
Description	<p>The control reference number in the trailer segment does not match that in the header segment.</p> <p>IMPORTANT: This number might also have been generated by the Reconciliation server, not the TRM. Refer to the MW Translator Operator Guide and Reference for more information. Specifically, refer to the topic "Logging and Reconciliation Event Messages" in the section, "Using Audit and Reconciliation," or simply search for the error number on the Index or Help tabs of the online help.</p>

2002 Trailer count does not match actual count

Text	Trailer count does not match actual count Includes pertinent information, such as, Actual count: 3 Trailer count: 2 Wrapper Name: UNB
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse
Description	<p>The number of entities (segments in a document, documents in a functional group or interchange, or functional groups in an interchange) does not match the corresponding value in the trailer segment.</p> <p>IMPORTANT: This number might also have been generated by the Reconciliation server, not the TRM. Refer to the MW Translator Operator Guide and Reference for more information. Specifically, refer to the topic "Logging and Reconciliation Event Messages" in the section, "Using Audit and Reconciliation," or simply search for the error number on the Index or Help tabs of the online help.</p>

2003 Undefined sending partner

Text	Undefined sending partner Includes pertinent information, such as, Partner ID: TEST-SEND-PAR Qual: ZZ
Error Action	Accept or Reject
Error Type	Compliance or configuration
Processing Phase	Parse
Description	The sending partner is not configured, but the Standard Id configuration specifies that sending partner is required. Add partner configuration or change Standard Identification Properties window to not require sending partner.

2004 Undefined receiving partner

Text	Undefined receiving partner Includes pertinent information, such as, Partner ID: TEST-REC-PAR Qual: ZZ
Error Action	Accept or Reject
Error Type	Compliance or configuration
Processing Phase	Parse
Description	The recipient partner is not configured, but the Standard Id configuration specifies that recipient partner is required. Add partner configuration or change Standard Id to not require recipient partner.

2005 Routing address is not defined for sending partner

Text	Routing address is not defined for sending partner Includes pertinent information, such as, Partner ID: TEST-SEND-PAR Qual: ZZ
Error Action	Accept or Reject
Error Type	Configuration
Processing Phase	Parse
Description	A location address could not be found to route information to the sending partner. Add a location value that can be used for the sending partner.

2006 Routing address is not defined for receiving partner

Text	Routing address is not defined for receiving partner Includes pertinent information, such as, Partner ID: TEST-REC-PAR Qual: ZZ
Error Action	Accept or Reject
Error Type	Configuration
Processing Phase	Parse
Description	A location address could not be found to route information to the receiving partner. Add a location value that can be used for the receiving partner.

2009 Invalid wrapper definition

Text	Invalid wrapper definition; no contents Includes pertinent information, such as, Wrapper Name: NULLWRAP Wrapper definition file: C:\Workbench\CFG\APPL\NULLWRAP.TXT
Error Action	Accept or Reject
Error Type	Compliance or configuration
Processing Phase	Parse
Description	Even null wrappers must define contents header segment. Fix wrapper definition in Wrapper window.

2010 Unable to find trade agreement

Text	Trade agreement not found on recipient partner or Standard ID
Error Action	Accept or Reject
Error Type	Configuration
Processing Phase	Parse
Description	The application is not configured to support the receipt of this document. A matching trade agreement (TA) must be configured for the partner relationship (if one was found), the recipient partner (if one was found) or the standard ID record. Furthermore, if the trade agreement is defined on the recipient partner or standard ID record and a relationship or recipient partner is defined, then the box Continue search for TA on Recipient Partner must be checked. Note that the error message states which of the three places were searched. If the place where the TA was

defined is not searched, it is because either the **Continue search for TA on Recipient Partner** box is not checked or the Partner Relationship or Recipient partner was not found.

The application attempts to match the incoming data as stored in the internal fields with the values defined on the **IDs** pages for partner IDs or on the **Matching Fields** page for trade agreement profiles. Remember that the key values listed on the processing report reflect the incoming values stored in the internal fields assigned to the wrapper elements. If the Field for a partner ID element is unassigned (blank or <None>), the value stored in the internal field for that element is null. To match a null value in an internal field, the configuration values must have no entry.

To correct the error, define a trade agreement for the partner relationship or recipient partner, or define a default TA for the Standard ID record that matches the keys reported on the processing report, or add the sender as a member of the recipient partner's group. If the trade agreement is part of a closed group (specified on list of trade agreements on the definition where the TA is found), the sender must be a member of the group (listed in the recipient group's **Members** folder). You will have to set options to **Always Print Reports** to see the key information on the processing report.

For an inbound XML document:

- **Specify the user exit, trmxml, on the User Exits tab of the Modify Options window** (on page 521).
- Use the pre-defined null wrapper for XML, XMLIN, which stores the document ID. You can copy this wrapper from the **XML-X12 example** (on page 368).
- For the Contents header in the wrapper definition, you must enter the segment tag that represents the XML root element, which should be the first segment in the document.
- Specify the user exit, trmxml, for the XMLIN wrapper as a **Pre-Process Method** on the **Wrapper Properties|General** tab.

2011 Invalid wrapper segment found

Text	Invalid wrapper segment found: UNQ Includes pertinent information, such as, Message: UNB^...
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse and Generate
Description	A segment was found in the wrapper that is not defined. For XML input, this may be because: <ul style="list-style-type: none"> ▪ The Xerces parser has not parsed the data. Make sure the trmxml user exit is

specified for the environment, and that the full path is correct. For MessageWay, the path to the file and the file must be owned by the owner of MessageWay, which is typically mway on UNIX/Linux machines.

- The tag of the root element in the input has not been stored in the Data Element Store. This happens when Xerces throws errors during validation in the pre-processing user exit, trmxml. Correct the XML errors and retest.

2012 Unable to find the destination location

Text	Unable to find the destination location
Error Action	Accept or Reject
Error Type	Configuration
Processing Phase	Parse
Description	There was no source or destination location defined for the outbound partner and no default provided in Standards Identification.

2013 Unable to identify the document

Text	Unable to identify the document defined in the wrapper Includes pertinent information, such as, Document Header name: Header segment Document Header Segment Tag: HDR Wrapper definition file: C:\Workbench\CFG\APPL\NULLWRAP.TXT
Error Action	Accept or Reject
Error Type	Configuration
Processing Phase	Parse
Description	No documents in wrapper or document begins with segment other than one defined as header segment in Contents area of Wrapper window.

2015 Defined or received delimiters are invalid

Text	Defined or received delimiters are invalid Includes pertinent information, such as, Seg Term: ~ Tag Delim: = Ele Delim: * Comp Delim: : Rel Char: =
Error Action	Abort
Error Type	Configuration or compliance

Processing Phase	Parse
Description	The segment terminator and the release character are the same or are the same as one of the other delimiters.

2016 Wrapper trailer is missing

Text	Wrapper trailer is missing. Includes pertinent information, such as, Wrapper: ISA
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Parse and Generate
Description	The interchange or functional group trailer is missing.

4001 Invalid control reference validation method

Text	Invalid control reference validation method: ONEOFF Includes pertinent information, such as, Sending Partner ID: TEST-SEND-PAR Qual: ZZ Recipient Partner ID: TEST-REC-PAR Qual: ZZ
Error Action	Abort
Error Type	Configuration
Processing Phase	Parse
Description	The control reference validation method configured for the specified partners is not valid. It must be either one of the pre-defined validation methods or a customer defined validation method, handled by a registered user exit. IMPORTANT: This number might also have been generated by the Reconciliation server, not the TRM. Refer to the MW Translator Operator Guide and Reference for more information. Specifically, refer to the topic "Logging and Reconciliation Event Messages" in the section, "Using Audit and Reconciliation," or simply search for the error number on the Index or Help tabs of the online help.

4002 Invalid control reference generation method

Text	Invalid control reference generation method: ONEOFF Includes pertinent information, such as, Sending Partner ID: TEST-SEND-PAR Qual: ZZ Recipient Partner ID: TEST-REC-PAR Qual: ZZ
Error Action	Abort
Error Type	Configuration
Processing Phase	Parse
Description	<p>The control reference generation method configured for the specified partners is not valid. It must be either one of the pre-defined generation methods supplied with the Workbench or a customer defined generation method, handled by a registered user exit.</p> <p>IMPORTANT: This number might also have been generated by the Reconciliation server, not the TRM. Refer to the MW Translator Operator Guide and Reference for more information. Specifically, refer to the topic "Logging and Reconciliation Event Messages" in the section, "Using Audit and Reconciliation," or simply search for the error number on the Index or Help tabs of the online help.</p>

4003 Duplicate control reference

Text	Duplicate control reference: 100000052
Error Action	Reject or Accept
Error Type	Compliance
Processing Phase	Parse
Description	<p>The received control reference has been previously processed.</p> <p>IMPORTANT: This number might also have been generated by the Reconciliation server, not the TRM. Refer to the MW Translator Operator Guide and Reference for more information. Specifically, refer to the topic "Logging and Reconciliation Event Messages" in the section, "Using Audit and Reconciliation," or simply search for the error number on the Index or Help tabs of the online help.</p>

4004 Control reference is out of sequence

Text	Control reference is out of sequence. Includes pertinent information, such as, Received control reference: 100000052 Expected control reference: 100000051
-------------	---

Error Action	Reject or Accept
Error Type	Compliance
Processing Phase	Parse
Description	<p>The received control reference is out of sequence. One or more prior interchanges must be processed first.</p> <p>IMPORTANT: This number might also have been generated by the Reconciliation server, not the TRM. Refer to the MW Translator Operator Guide and Reference for more information. Specifically, refer to the topic "Logging and Reconciliation Event Messages" in the section, "Using Audit and Reconciliation," or simply search for the error number on the Index or Help tabs of the online help.</p>

4005 Control reference overflow

Text	<p>Control reference overflow</p> <p>Includes pertinent information, such as, Last control reference: ABCDE9999</p>
Error Action	Abort
Error Type	Configuration
Processing Phase	Parse
Description	<p>Control reference generation failed due to overflow. Either the maximum width (14) for the control references has been exceeded, or overflow occurred in the sequence part of a 5SUB4 control reference.</p> <p>IMPORTANT: This number might also have been generated by the Reconciliation server, not the TRM. Refer to the MW Translator Operator Guide and Reference for more information. Specifically, refer to the topic "Logging and Reconciliation Event Messages" in the section, "Using Audit and Reconciliation," or simply search for the error number on the Index or Help tabs of the online help.</p>

4006 Inconsistent control reference generation

Text	<p>Generation type is inconsistent with last control reference sent.</p> <p>Includes pertinent information, such as, Last Control Reference: ABCDEFGHI Generation Type: ONEUPN</p>
Error Action	Abort
Error Type	Configuration
Processing Phase	Parse

Description Attempting to generate a one-up numeric control reference when the last generated control reference is alphanumeric.

IMPORTANT: This number might also have been generated by the Reconciliation server, not the TRM. Refer to the MW Translator Operator Guide and Reference for more information. Specifically, refer to the topic "Logging and Reconciliation Event Messages" in the section, "Using Audit and Reconciliation," or simply search for the error number on the **Index** or **Help** tabs of the online help.

4007 Control reference number is in use

Text Record with this control reference number is in use
Includes pertinent information, such as,
Table Name: PARCTRL Control ref. number: 100000052

Error Action Abort

Error Type Configuration

Processing Phase Parse

Description The same control reference partner combination is in use by a concurrent process.

4008 Received control reference is blank

Text Received control reference is blank

Error Action Reject or accept

Error Type Compliance

Processing Phase Parse

Description The received control reference is missing or spaces.

5001 Invalid user exit name

Text Invalid user exit name: MYEXIT

Error Action Abort

Error Type Configuration

Processing Phase Parse

Description An Edibasic User() function has been called with the name of an unregistered user exit.

6001 Too few segment or loop repeats

Text	Too few segment or loop repeats
Error Action	Accept or reject
Error Type	Compliance
Processing Phase	Generate
Description	This exception should never occur. You would instead get error 1002 (Mandatory element missing). Users can enter a negative number of repeats, which should not be possible, and then you might get it. Otherwise, the minimum is zero or greater than zero, which is a result of the requirement designator (req), which is not zero when req is M (mandatory) or zero when req is O (optional) or C (conditional).

6002 Too many segment repeats

Text	Too many segment repeats Includes pertinent segment information, such as, Segment: LIN(47) Segment count: 839 Occurrences: 1000 Maximum allowed: 999
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Generate
Description	The segment has more than the maximum number of occurrences defined for the standard.

6003 Mandatory entity missing

Text	Mandatory loop/segment/element/composite missing Includes pertinent information, such as, Segment: LIN(47) Segment count: 32 Composite Seq: 6 Composite ID: C186 Element Seq: 2 Element ID: 6060
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Generate
Description	Missing mandatory loop, segment, element, or composite.

6005 Too many loop repeats

Text	Too many loop repeats Includes pertinent information, such as, Loop: LIN{47} Segment Count: 837 Occurrences: 1000 Maximum allowed: 999
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Generate
Description	The loop has more than the defined maximum occurrences.

6009 Too many omitted elements

Text	Too many omitted elements. The maximum number of consecutive omitted elements allowed is 256.
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Generate
Description	There are more than 256 consecutive elements omitted in the segment.

6010 Reject during generate; processing aborted

Text	Reject during generate; processing aborted. Usually followed by another error, such as those in the 9000 series.
Error Action	Abort
Error Type	Configuration
Processing Phase	Generate

Description	<p>A system error caused the abort during the generation phase. The error might be caused by any one of the following types of errors:</p> <ul style="list-style-type: none">▪ File open▪ File read▪ File name not configured▪ User defined exception with invalid exception #▪ Forced Interchange Reject▪ Configuration file is empty▪ Unidentified standard▪ Unable to load document definition file▪ Invalid condition while loading document definition▪ Unable to load wrapper definition file▪ Unable to load map definition file▪ Invalid element map type; map is not in sync with document▪ Invalid composite map type; map is not in sync with document▪ Inbound document/wrapper is not the same as input document/wrapper defined in map
--------------------	---

6011 Error during acknowledgment generation; processing aborted

Text	<p>An error occurred during acknowledgement generation; processing aborted Usually followed by another error, such as those in the 9000 series.</p>
Error Action	Abort
Error Type	Configuration
Processing Phase	Generate

Description	<p>A system error caused the abort during the generation phase of an acknowledgment. The error might be caused by any one of the following types of errors:</p> <ul style="list-style-type: none"> ▪ File open ▪ File read ▪ File name not configured ▪ User defined exception with invalid exception # ▪ Forced Interchange Reject ▪ Configuration file is empty ▪ Unidentified standard ▪ Unable to load document definition file ▪ Invalid condition while loading document definition ▪ Unable to load wrapper definition file ▪ Unable to load map definition file ▪ Invalid element map type; map is not in sync with document ▪ Invalid composite map type; map is not in sync with document ▪ Inbound document/wrapper is not the same as input document/wrapper defined in map
--------------------	---

6102 Data element is too long

Text	<p>Data element is too long</p> <p>Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "374382938473823423"</p>
Error Action	Accept or Reject
Error Type	Compliance error.
Processing Phase	Generate
Description	Number of characters in the element is greater than what is allowed by the element definition.

6103 Data element is too short

Text	<p>Data element is too short</p> <p>Includes pertinent information, such as, Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "3"</p>
Error Action	Reject

Error Type	Compliance error.
Processing Phase	Generate
Description	Number of characters in the element is less than what is allowed by the element definition.

6104 Data element contains invalid code value

Text	Data element contains invalid code value Includes pertinent information, such as, for Part 1 (if element uses partitions, such as X12 element 103) Segment: DTM(24) Segment count: 10 Element Seq: 5 Element ID: 624 Element Value: "3", or Element's Part 1 value: AAA (if element uses partitions, such as X12 element 103)
Error Action	Reject
Error Type	Compliance error.
Processing Phase	Generate
Description	You have chosen to validate element ID codes, and the referenced code does not match any of those listed for the specified element or for the specified partition of the element.

6201 Composite conditional failed

Text	Composite conditional failed Includes pertinent information, such as, Exception # 1201 (Reject) - Composite conditional failed Segment: HI(56) Segment count: 50 Composite Seq: 1 Composite ID: C022 Cond: 1 P 03 04
Error Action	Accept or Reject
Error Type	Compliance
Processing Phase	Generate

Description The identified composite element has a condition of a specific type, in this case, paired (P), which means when one of the elements identified in the condition is present, all must be present. The condition number (1 P 03 04) is the number used by the standard to identify this condition. For X12, this notation means the first condition (1) on this composite is a paired condition (P) and both component elements 3 (03) and 4 (04) must be present.

6202 Composite conditional failed

Text Composite conditional failed
Includes pertinent information, such as,
Exception # 1202 (Reject) - Composite conditional failed
Segment: HI(56) Segment count: 50
Composite Seq: 1 Composite ID: C022
Cond: 2 E 08 09

Error Action Accept or Reject

Error Type Compliance

Processing Phase Generate

Description The identified composite element has a condition of a specific type, in this case, paired (E), which means when one of the elements identified in the condition is present, all must be present. The condition number (2 E 08 09) is the number used by the standard to identify this condition. For X12, this notation means the second condition (2) on this composite is an exclusion condition (E) where elements 8 (08) and 9 (09) are mutually exclusive. When 8 is present, element 9 must not be, and when 9 is present, 8 must not be present.

6301 Segment conditional failed

Text Segment conditional failed
Includes pertinent information, such as,
Segment: MEA(21) Segment count: 3
Cond: 2 C 03 04

Error Action Accept or Reject

Error Type Compliance

Processing Phase Generate

Description The identified segment has a condition of a specific type, in this case, conditional (C), which means that when the first element is present, all elements identified in the condition must be present. The condition number (2 C 03 04) is the number used by the standard to identify this condition. For X12, this notation means that if element 3 is present, then element 4 must also be present.

6305 Segment conditional failed

Text	Segment conditional failed Includes pertinent information, such as, Segment: MEA(21) Segment count: 3 Cond: 6 E 08 03
Error Action	Accept or Reject
Error Type	Compliance error
Processing Phase	Generate
Description	The identified segment has a condition of a specific type, in this case, Exclusion (E), which means that at least one of the elements identified in the condition must be present. The condition number (6 E 08 03)) is the number used by the standard to identify this condition. For X12, this notation means the sixth condition (6) on this segment is an exclusion condition (E) where elements 3 and 8 are mutually exclusive. If 3 is present, 8 should not be, or if 8 is present, 3 should not be.

6401 Invalid Bounded Loop Trailer Segment

Text	Invalid Bounded Loop Trailer Segment Includes pertinent information, such as, Loop Id Expected: 0100 Loop Id Found: 0200
Error Action	Accept or Reject
Error Type	Compliance error.
Processing Phase	Generate
Description	Found bounded loop trailer, but the loop ID does not match the ID in the loop header.

6501 An error occurred during generation; output rolled back

Text	An error occurred during generation; output rolled back
Error Action	Warning
Error Type	System
Processing Phase	Generate

Description If a reject occurs during wrapper header generate or document generate, then document-level reject is attempted by calling rollback on each generated output. If a rollback is successful this exception is generated. The status of the Functional Group (FG) and Interchange is set to Reject (R). Processing continues with the remaining documents in the Interchange or remaining FG's in the Interchange or moves on to the next interchange depending at what wrapper level the reject occurred.

6502 An error occurred during generation; output aborted

Text An error occurred during generation; output aborted

Error Action Abort

Error Type System

Processing Phase Generate

Description If a rollback was not possible for a given output, then the output is aborted and deleted.

6503 The Interchange is rejected

Text The Interchange is rejected.

Error Action Reject

Error Type System

Processing Phase Generate

Description If an error occurs in the Interchange trailer, the Interchange is rejected. All lower-level Functional Group and Document acknowledgements are also deleted.

6504 The Functional Group is rejected

Text The Functional Group is rejected

Error Action Reject

Error Type System

Processing Phase Generate

Description If an error occurs in the trailer, the Functional Group is rejected. All document-level acknowledgements are also deleted.

6505 An error occurred during rollback; the Interchange is rejected

Text	An error occurred during rollback; The Interchange is rejected.
Error Action	Reject
Error Type	System
Processing Phase	Generate
Description	If a problem occurs during rollback because a checkpoint was not done or some file IO errors occurred, then all outputs are aborted and deleted and this exception is generated. The entire Interchange is rejected.

7001 Unable to Find Segments in the Document

Text	Unable to find segments in the document
Error Action	Reject
Error Type	Configuration
Processing Phase	Parse
Description	<p>The incoming document header does not match the configuration of the source document specified in the trade agreement profile.</p> <p>For XML input, this may be because:</p> <ul style="list-style-type: none">▪ The tag of the root element in the input data does not match the Description of the first segment of the document. This is because the pre-processing user exit, trmxml, stores the tag of the root element in the Document ID internal field. MW Translator matches the Document ID value stored in the internal field to the value of the Description for the first segment of the document specified on the trade agreement.▪ When using namespaces, the namespace specified for a segment may be missing the prefix. Add the prefix to a segment prior to where the namespace is used, typically on the first segment of the document, which represents the root element in the XML input. The MW Translator segment must use a category of XML to display the XML tab on which you type the namespace using the following syntax: <code>xmlns[:<i>prefix</i>]="<i>URI</i>"</code>▪ When namespaces are defined for the first segment of a document, which represents the xml root element, the input xml data does not include the namespace designation. You should remove the namespace from the XML tab of the Segment Properties window. Make sure you remove everything, including any hidden carriage returns.

9001 Processing aborted by user

Text	Processing aborted by user.
Error Action	Abort
Error Type	User-initiated
Processing Phase	Parse
Description	The user has manually canceled processing.

9003 Unable to identify standard of message

Text	Unable to identify standard of message from: UNKNOWN Includes pertinent information, such as, Message: ISA*...
Error Action	Abort
Error Type	System
Processing Phase	Parse
Description	<p>The incoming data cannot be identified from the defined Standard Identification parameters. Make sure the incoming wrapper is correctly defined and is entered on the list of one of the locations in the Locationes/StdID folder of the Partner Explorer window. If you use them, make sure the matching criteria on the General page are correct. If the standard wrapper is defined in a location other than <Default>, make sure the incoming location is correct. If the location is blank or does not exist, the TRM attempts to match the criteria for the wrappers listed only in the <Default> location.</p> <p>For files containing multiple interchanges, make sure you have correct matching criteria for the interchanges following the first one, under the following conditions:</p> <ul style="list-style-type: none">▪ When a user exit has modified wrapper information for each interchange, for example by using the AddTag user exit to add a wrapper to each interchange▪ For a fixed-length file, where the IO Mode is defined as Text on the first interchange wrapper, and where the end-of-segment marker is two characters, such as carriage-return/line-feed (CRLF), and when you use offset values to identify matching criteria beyond the first segment <p>Under the first condition, the first attempt to identify the standard occurs before the user exit adds the wrappers to each of the interchanges. When it identifies the remaining interchanges, you might need a second standard ID with different matching criteria the wrapper segments added by the user exit.</p> <p>Under the second condition, the TRM may itself have modified the input data by replacing line markers with its own internal value. Once the TRM determines the definition of the first incoming wrapper from the standard identification process, it</p>

re-opens the input based on the standard and wrapper definitions. When the IO Mode is set to Text, the TRM replaces the carriage-return/linefeed (CRLF), linefeed (LF) or record break with a single new-line (NL) character for this and any subsequent interchanges in this file. For IO Mode, the TRM only uses the setting for the first interchange wrapper in the file. Therefore, when you have multiple interchanges in a file and the input line control is more than one character, you must be sure that the offset for the matching criteria for subsequent interchanges allows for the change in segment terminators from two (CRLF) to one (NL) character.

9004 Unable to open file

Text	Unable to open Config/Partner/Standards Id/Control Reference/Transaction Set/Wrapper Definition/Wrapper Map/Map/Cross-reference/Profile file Includes pertinent information, such as, File name: C:\Workbench\CFG\PROFILE.TXT Error: No such file or directory
Error Action	Abort
Error Type	System
Processing Phase	Parse
Description	Unable to open a system configuration file - usually because it is not found. Generate the required file and retry.

9005 File read error

Text	File read error on Config/Partner/Standard Id./Control Reference/Transaction Set/Wrapper Definition/Wrapper Map/Map/Cross-reference/Profile file
Error Action	Abort
Error Type	System
Processing Phase	Parse
Description	System read error on system configuration file.

9006 Filename not configured in INI file

Text	The file name is not configured in the INI file
Error Action	Abort
Error Type	System
Processing Phase	Parse

Description Missing file name not assigned for system file in EDI configuration file (INI file). This file is no longer used for the TRM on the Workbench.

9007 User defined exception with invalid exception

Text User defined exception with invalid exception #

Error Action Abort

Error Type System

Processing Phase Parse

Description Refer to the user documentation.

9008 Forced Interchange Reject

Text Forced Interchange Reject

Error Action Abort

Error Type System

Processing Phase Parse

Description

9009 Configuration file is empty

Text Config/Partner/Standard ID/Control Reference/Document Definition/Wrapper Definition/Wrapper Map/Map/Cross Reference/Profile/ParRel/Unknown file configuration file is empty.
Includes pertinent information, such as,
FILE123.TXT

Error Action Abort

Error Type System

Processing Phase Parse

Description The generated text file may have been deleted. Generate all files and retest.

9010 Unable to load set definition file

Text	Unable to load set definition file Includes pertinent information, such as, File name: C:\Workbench\CFG\X12\X850.TXT
Error Action	Abort
Error Type	System
Processing Phase	Parse
Description	Problem loading document definition file. See error 9006. Text file may be corrupt or missing. Regenerate file.

9011 Invalid condition while loading set

Text	Invalid condition while loading set Includes pertinent information, such as, Condition: Z
Error Action	Abort
Error Type	System
Processing Phase	Map
Description	Invalid segment condition in document text file. Document text file is corrupt. You must regenerate.

9019 TRM failure

Text	The text for this error varies. This is a list of some of the messages showing the error constant, error text and return code: <ul style="list-style-type: none"> ▪ TRM_FAILURE: TRM failure -2 ▪ TRM_INVALID_CFG_FILENAME: Invalid INI filename -6 ▪ TRM_INVALID_MISSING_CFGFILE: Missing INI file -7 ▪ TRM_FUNC_NULL: User exit function pointer is NULL -8 ▪ TOO_MANY_AUDIT_FUNCS: Too many audit functions registered -9 ▪ TRM_INVALID_MISSING_DLL: Missing or Invalid User Exit DLL -42
Error Action	Abort
Error Type	System
Processing Phase	NA

Description The MW Translator service has received an error from the TRM. They typically relate to initialization problems with the TRM configuration file, trm.ini, or user exits.

9020 Unable to load map definition file

Text Unable to load map definition file
Includes pertinent information, such as,
File name: C:\Workbench\CFG\MAP\MYMAP.TXT

Error Action Abort

Error Type System

Processing Phase Map

Description Problem loading map file. See error 9011. File may be corrupt or missing. Regenerate map file.

9021 Invalid element map type

Text Invalid element map type; map is not in sync with document
Includes pertinent information, such as,
Map Type: ONCE

Error Action Abort

Error Type System

Processing Phase Map

Description Map file is not in sync with document file. If you have changed your output file definition after you created the map, you must reload the destination document using the Map window, make sure all of your mappings are still correct, and regenerate the map. Regenerate document. Set Options to **Print File Names** on translation report to determine which map and document were used.

9022 Invalid composite map type

Text Invalid composite map type; map is not in sync with document
Includes pertinent information, such as,
Map Type: LIT

Error Action Abort

Error Type System

Processing Phase Map

Description Map file is not in sync with document file. If you have changed your output file definition after you created the map, you must reload the destination document using the Map window, make sure all of your mappings are still correct, and regenerate the map. Regenerate document. Set Options to **Print File Names** on translation report to determine which map and document were used.

9023 Inbound wrapper/document is not the same as input wrapper/document defined in map

Text Inbound wrapper/document is not the same as input wrapper/document defined in map
Includes pertinent information, such as,
Map: MYMAP
Map input document: MYSTD, 1, MYPO

Error Action Warning

Error Type System

Processing Phase Map

Description The document used to parse the input is not the same as the document specified as the source document in the map.

9024 Inbound Wrapper Is Not the Same as Input Wrapper Defined in Map

Text Inbound wrapper is not the same as input wrapper defined in map
Includes pertinent information, such as,
Map: NewMap
Map input wrapper: Example, 1.0, MyWrap

Error Action Warning

Error Type System

Processing Phase Map

Description Map file is not in sync with wrapper file. If you have changed your input wrapper definition after you created the map, you must reload the source wrapper using the Map window, make sure all of your mappings are still correct, and regenerate the map. Set Options to **Print File Names** on translation report to determine which map and document were used.

9030 Unable to load wrapper definition file

Text	Unable to load wrapper definition file Includes pertinent information, such as, File name: C:\Workbench\CFG\APPL\MYWRAP.TXT
Error Action	Abort
Error Type	System
Processing Phase	Map
Description	Problem loading wrapper text file. See error 9022. File may be corrupt or missing. Regenerate map file.

9040 Invalid Instruction

Text	Invalid Instruction; corrupt configuration file Includes pertinent information, such as, OpCode: JMPZ
Error Action	Abort
Error Type	System
Processing Phase	Map
Description	Map, document or wrapper definition file is corrupt. Regenerate the file.

9050 File version mismatch

Text	File Version Mismatch Includes pertinent information, such as, File Version: 0.4 Run-Time Module Version 1.0 File name: C:\Workbench\CFG\PARTNER.TXT
Error Action	Abort
Error Type	System
Processing Phase	Map
Description	Version of configuration file is not compatible with Translator Runtime Module. Regenerate file with compatible version of Workbench.

9060 Missing Profile

Text	Missing Profile: MYPROFILENAME
Error Action	Warning
Error Type	System
Processing Phase	Map
Description	Missing Trade Agreement or Acknowledgment Profile. Profile name specified in Partner or Standards Identification is not present in Profile file. This can happen if a Trade Agreement or Acknowledgment profile is deleted and is still referenced. This is just a warning. This may cause exception # 2010 later on.

9070 Unable to open control reference database

Text	Unable to open control reference database
Error Action	Abort interchange
Error Type	System
Processing Phase	Parse or generate
Description	

9080 Unable to open control reference table

Text	Unable to open control reference table
Error Action	Abort interchange
Error Type	System
Processing Phase	Parse or generate
Description	

9081 Unable to read control reference database

Text	Unable to read control reference database
Error Action	Abort interchange
Error Type	System
Processing Phase	Parse or generate

Description

9082 Unable to write to the control reference database

Text	Unable to write to the control reference database
Error Action	Abort interchange processing with failure. Do not produce acknowledgment. Report translation failure.
Error Type	System
Processing Phase	Generate
Description	

9083 Unable to update control reference database

Text	Unable to update control reference database
Error Action	Abort interchange processing with failure. Do not produce acknowledgment. Report translation failure.
Error Type	System
Processing Phase	Generate
Description	

9090 Invalid document/wrapper/map definition file

Text	Invalid document/wrapper/map definition file Includes pertinent information, such as, File Name: C:\Workbench\CFG\APPL\MYWRAP.TXT
Error Action	Abort
Error Type	System
Processing Phase	Parse, map or generate
Description	

9100 System Memory Error

Text	System memory error.
Error Action	Abort

Error Type	System
Processing Phase	Parse, map or generate
Description	There has been a failure to allocate enough memory during translation. Try to increase the amount of virtual memory available to the process or decrease the size of the input file.

9101 Fatal System Exception

Text	Fatal System Exception Exception: MWrapper::parse() failed – or – Exception: MDocWrapper::translate() failed
Error Action	Abort
Error Type	System
Processing Phase	Parse or map
Description	This may be caused by errors related to XML namespace definitions. When using a namespace in an input XML document, the namespace must be defined on a segment before the place it is actually used. Typically, we define the namespaces on the first segment of the document, which represents the XML root element. An error in the syntax of the namespace will cause this error. Make sure you use the following syntax for the namespace: <code>xmlns[:prefix]="URI"</code> To remove a namespace definition, you must remove the namespace from the XML tab before you change the Category to blank and then generate the document, in case you decide to do this. Changing the category to blank, which hides the XML tab, does not remove the namespace.

9110 Security Reject

Text	Unable to find required security document DOCNAME in Trade Agreement Profile: TAPROFILE NAME
Error Action	Reject interchange
Error Type	System
Processing Phase	Parse

Description The TRM did not find the security document identified in the Trade Agreement Profile window, Options page, used for this incoming interchange. The wrapper level status will be **S**.

9120 Unable to load user exit DLL

Text Unable to load user exit DLL
Includes pertinent information, such as,
DLL File Name: C:\Workbench\CFG\ADDTAG.DLL

Error Action Reject interchange

Error Type System

Processing Phase Parse

Description The user exit file could not be found. Check the file name and location of the file.

9130 Unable to register user exit DLL

Text Unable to register user exit DLL
Includes pertinent information, such as,
DLL File name: C:\Workbench\CFG\ADDTAG.DLL

Error Action Reject interchange

Error Type System

Processing Phase Parse

Description The user exit was found, but it could not be registered. Check to make sure this file contains a valid DLL (Windows) or SO (UNIX/Linux).

9140 Security Reject

Text Unable to validate required security document DOCNAME in Trade Agreement Profile: TAPROFILE NAME

Error Action Reject interchange

Error Type System

Processing Phase Parse

Description The TRM did find the security document identified in the Trade Agreement Profile, but the processing of that document did not result in the security status being set, for example, when **ERMSecurityStatus()** was not called. Both the security document status and the wrapper level status will be **S**.

9150 Security Abort

Text	Invalid security user exit name: AAA in Acknowledgment Profile: CONTRLNoUXit, or Invalid security user exit name: BBB in Trade Agreement Profile: TestTA
Error Action	Abort interchange
Error Type	System
Processing Phase	Parse
Description	<p>For the acknowledgment, this was caused because the security user exit 'AAA' was defined in the ack profile CONTRLNoUXit, but no such user exit was registered. Possible causes: a) the name ('AAA') is wrong, b) The user exit is not loaded (DLL for Windows, SO for UNIX/Linux). The same reasoning applies to this error for a trade agreement.</p> <p>For the Trade Agreement profile, the invalid security user exit may either be defined on the Trade Agreement Profile, Input tab or on the Trade Agreement Output Properties, Summary Document tab.</p> <p>Both the security document status and the wrapper level status will be S.</p>

10200 Internal System Error

Text	Internal system error.
Error Action	Abort
Error Type	System
Processing Phase	Parse, map or generate
Description	An unknown internal program error occurred. Contact your support system.

15001 Security Reject

Text	Security Exception. FOLLOWED BY A MESSAGE FROM A SECURITY USER EXIT
Error Action	Reject interchange
Error Type	System
Processing Phase	Parse

Description The validation user exit returned a reject response. That is, **ERMSecurityStatus()** was called with a status of **ACT_SEC_REJECT**. This is the user exit defined in the validation method of the security document. The security document name is in the **Security Document** field on the Trade Agreement Profile window, **Options** tab. Both the security document (e.g., AUTACK) status and the wrapper level status will be **S**.

20001 or higher User defined exception

Text User defined exception
 Might include additional information

Error Action Accept, Reject, or Abort


Error Type User-defined

Processing Phase Parse

Description These exceptions are generated by Edibasic code entered by the user.

Understanding the Processing Report

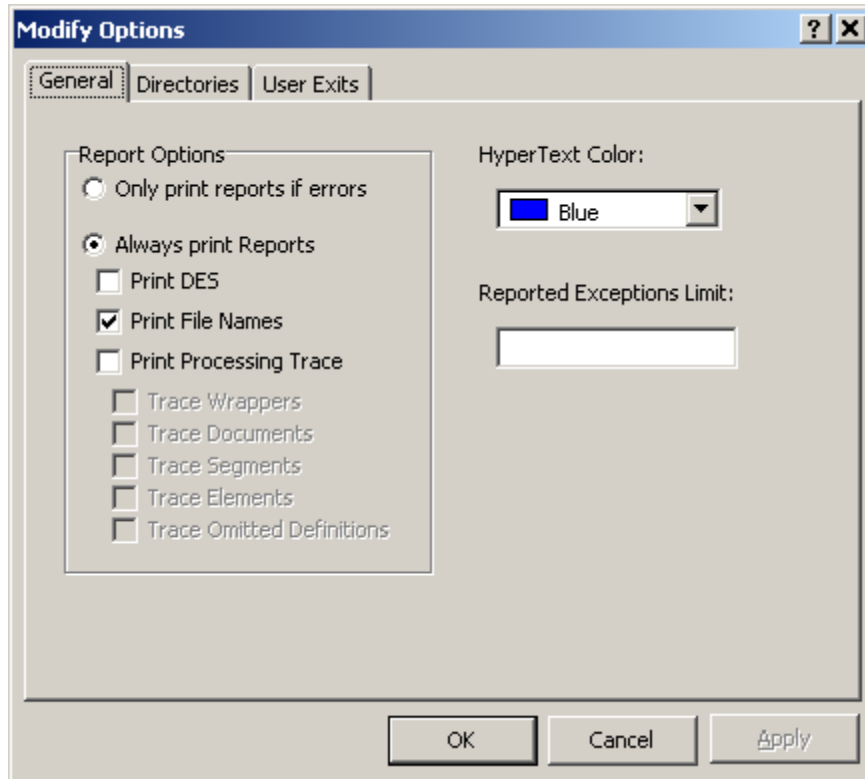
You can configure the TRM to always print a processing report or to print one only when there are errors. You will probably always want to print a report for testing, but for production, you will want to only print one if there are errors. When you select **Only print report if errors**, reports are generated only when there are errors or when there are print statements in a map.

In order to always generate a processing report, you must select **Always Print Reports** on the **General** tab of the Modify Options window. You access this window from the **File** menu by selecting **Options** or from the toolbar by selecting the **Options** button .

If you always want a report, you can display the following types of information:

Report Option	Type of Information
Always Print Reports	Basic processing information and status of input wrappers and documents resulting from compliance check
Print DES	Content of Data Element Store (DES), which is an internal table of data resulting from the parsing process
Print File Names	Full path and names of generated configuration and map files MW Translator uses.

Report Option	Type of Information
Print Processing Trace	Statements showing processing steps MW Translator uses as it parses and generates wrappers and documents, showing the parsed data before it is stored in the DES and data generated after mapping.



To print a report, you should *set the report options and run a test* (on page 185).

For more information about how to read a processing report, refer to the topic, *Reviewing Your Processing Report* (on page 394).

Status Information

The **Always Print Reports** option displays a code indicating the status of the documents and various wrapper levels. Wrapper level 1 is an interchange wrapper. Wrapper level 2 is a functional group wrapper.

The statuses that will print on the report are as follows:

Status Value	Description
A	Accept, with or without errors

Status Value	Description
R	Reject, continue processing if possible, otherwise abort
S	Security reject, continue processing

Consider the following status lines from a processing report.

```
Document Status: A Control Reference: 0003
Wrapper level 2 Status: A Control Reference: 100010001
```

Data Element Store Information

The **Print DES** displays a list of the values stored in the Data Element Store (DES). These values are placed here because of the parsing process. This information is separate from the internal field information used to match against configuration definitions.

Consider the following lines of information from the DES.

```
Dump of Data Element Store: 850
Segment: ST(1)
 1 '850'
 2 '0001'
```

File Name Information

The **Print File Names** option displays the text files the TRM used during processing.

Consider the following lines of information showing these files.

```
>>> Wrapper set definition file name:
C:\MWTranslator\DefaultEnv\Cfg\appl\W-EXAMPLE-1-FWRAP.txt <<<
>>> Map file name: C:\MWTranslator\DefaultEnv\Cfg\map\M-EXAMPLE-FWRAP.txt <<<
>>> Document definition file name:
C:\MWTranslator\DefaultEnv\Cfg\appl\D-EXAMPLE-1-FPO.txt <<<
>>> Map file name: C:\MWTranslator\DefaultEnv\Cfg\map\M-EXAMPLE-X850.txt <<<
Translation Output stream created: C:\MWTranslator\DefaultEnv\Out\BU08MUO.txt
Output number: 1 Source: TESTSEND-MAILBOX Destination: TESTREC-MAILBOX

>>> Control reference file name: C:\MWTranslator\DefaultEnv\Cfg\CTRLREF.TXT <<<
Map Name: EXAMPLE-X850
```

Processing Trace Information

The **Print Processing Trace** option and at least one of the sub-options displays the steps MW Translator uses as it searches for, parses and finally generates data. You can select the level of detail you want to see.

The following is a section of parse comments that could appear on the report. The data represented here is the form it has after parsing but before it is stored in the Data Element Store (DES). To view the data in the DES, you should select the option, **Print DES**, also.

```

...Parsing document: X12, EXAMPLE, 850...
...Looking for segment 'ST'
...Parsing segment 'ST' (1)...
...Parsed element seq 1   Element value: "850"
...Parsed element seq 2   Element value: "0001"
...Looking for segment 'BEG'
...Parsing segment 'BEG' (2)...
...Parsed element seq 1   Element value: "00"
...Parsed element seq 2   Element value: "NE"
...Parsed element seq 3   Element value: "PO12345"
...Parsed element seq 4   Element value: ""
...Parsed element seq 5   Element value: "950105"
...Parsed element seq 6   Element value: ""
...Looking for segment 'NTE'

```

The next is a section of generate comments that could appear on the report. The data represented here is the form it has after mapping.

```

...Output 1: Checking map for segment '***'...
...Generating output 1, segment '***' (1)...
...Generating output 1, element seq 1...   Element value: "****"
...Generating output 1, element seq 2...   Element value: "TESTSEND"
...Generating output 1, element seq 3...   Element value: "FPO"
Map Name: EXAMPLE-X850

...Generating output 1 document: EXAMPLE, 1, FPO with map: EXAMPLE-X850...
...Output 1: Checking map for segment 'HDR'...
...Generating output 1, segment 'HDR' (1)...
...Generating output 1, element seq 1...   Element value: "HDR"
...Generating output 1, element seq 2...   Element value: "PO12345"
...Generating output 1, element seq 3...   Element value: "051994"
...Generating output 1, element seq 4...   Element value: "1018"
...Generating output 1, element seq 5...   Element value: ""
...Output 1: Checking map for segment 'BUY'...
...Generating output 1, segment 'BUY' (2)...
...Generating output 1, element seq 1...   Element value: "BUY"
...Generating output 1, element seq 2...   Element value: "DATA ELEMENT CODE"

```

Parse Errors

Most parsing errors result from invalid input. In order to determine what valid input must be, you should know how the TRM parses data. The TRM parses, validates and processes data sequentially. The basic sequence is as follows:

- Parse and validate wrapper header segments
- Parse and validate document segments
- Parse and validate wrapper trailer segments

Here is how MW Translator handles some basic exceptions during parsing:

Location of Exception	Behavior
Document	Accept or reject the document, depending on configurations, and continue with the next document in the interchange.
Wrapper	Attempts to roll back processing prior to the parsing of the wrapper and skips the entity to continue with the next one. It progresses from the most specific level, document, to the least specific, interchange. That level will be accepted or rejected, depending on configurations. Any output or acknowledgments generated to this point specific to this level will be aborted.

You can view the input data in two states:

- After parsing, but before it is stored in the Data Element Store (DES)
- After it is stored in the DES

Parsing Process

The TRM processes data a segment at a time. As it parses header wrappers, it assumes that any unidentified segment is part of the content of the document. To determine when it has reached the end of the document it looks for one of the following:

- A document trailer segment
- Another valid wrapper segment
- Another instance of the document header segment

The TRM must also be able to distinguish between the end of the document and the beginning of a new interchange. This is not a problem for delimited contents, which use segment terminators and controlling wrapper structures. However, it can be a problem for fixed-length contents.

IMPORTANT: For fixed-length messages, the segment terminator must be a newline (N/L) character, achieved by setting **I/O Mode** to **Text** in the Wrapper Properties window. You must also assure that at least one of the three criteria listed above applies.

Viewing the Parsed Input Data

To view the input data after it has been parsed, but before it is generated, you must select the **Print Processing Trace** and **Trace Elements** from the Modify Options window.

The following is an example of parse information that would appear on the report.

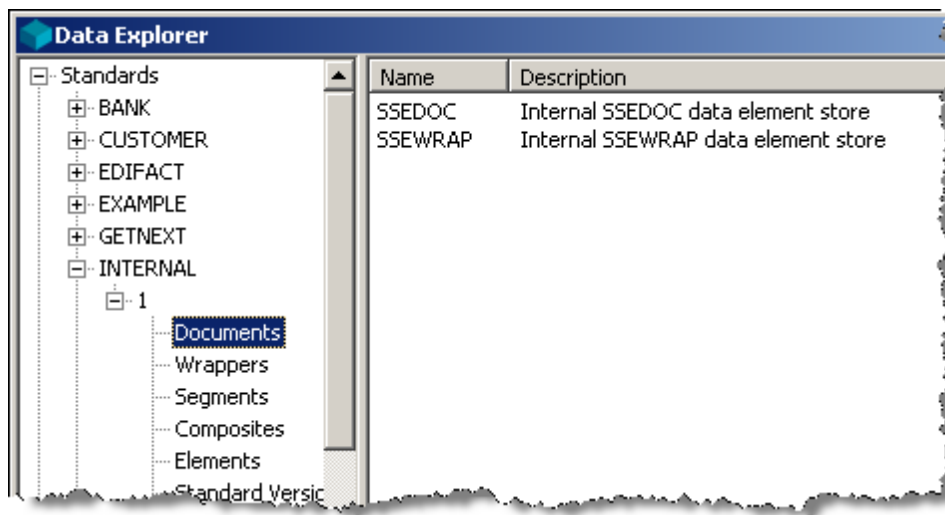
```
...Parsing document: X12, EXAMPLE, 850...
...Looking for segment 'ST'
...Parsing segment 'ST' (1)...
...Parsed element seq 1   Element value: "850"
...Parsed element seq 2   Element value: "0001"
...Looking for segment 'BEG'
...Parsing segment 'BEG' (2)...
...Parsed element seq 1   Element value: "00"
...Parsed element seq 2   Element value: "NE"
...Parsed element seq 3   Element value: "PO12345"
...Parsed element seq 4   Element value: ""
...Parsed element seq 5   Element value: "950105"
...Parsed element seq 6   Element value: ""
...Looking for segment 'NTE'
```


Viewing the Contents of the Data Element Store

When you view the processing report, you will see the results of the parsing process as it stored the information in the Data Element Store (DES). The DES shows the following contents:

- Contents parsed from the document (DOC)
- Contents parsed from the wrapper (WRAP)
- Status, statistics and error information for the document (SSEDOC)
- Status, statistics and error information for the wrapper (SSEWRAP)

The definitions for the document and wrapper are in the appropriate standards folder. You can view the definitions for the structure of the status statistics and errors information in the **Standards** folder in Data Explorer, by viewing the documents SSEDOC and SSEWRAP in the **Internal** folder:



For more information about the DES, view the following topics:

- *Understanding the Context List in the Data Element Store* (on page 186)
- *Level in DES Versus Level in Document Configuration* (on page 188)
- *Understanding Levels in the Context List* (on page 189)

Example of Document Data on Processing Report

The processing report displays the data parsed from the input document, including the document header. In the following example, the input document is an X12 850, purchase order. The information follows the information used to find the trade agreement, which it used to find the document definition, which it used to parse the document. The segment and element sequence numbers match those on the document and segment definitions, so it is easy to see when segments or elements are missing.

The following example is from the EXAMPLE-X850 translation included with the Workbench.

```
>>> Document definition file name:
C:\MWTranslator\DefaultEnv\Cfg\x12/D-X12-EXAMPLE-850.txt <<<
Trade Agreement found on Partner Relationship: EXAMPLE-X850
Identified document: 850 Version: 003030 Agency: X
Control Reference: 0001

Dump of Data Element Store: 850
Segment: ST(1)
  1 '850'
  2 '0001'
Segment: BEG(2)
  1 '00'
  2 'NE'
  3 'PO12345'
  5 '950105'
```

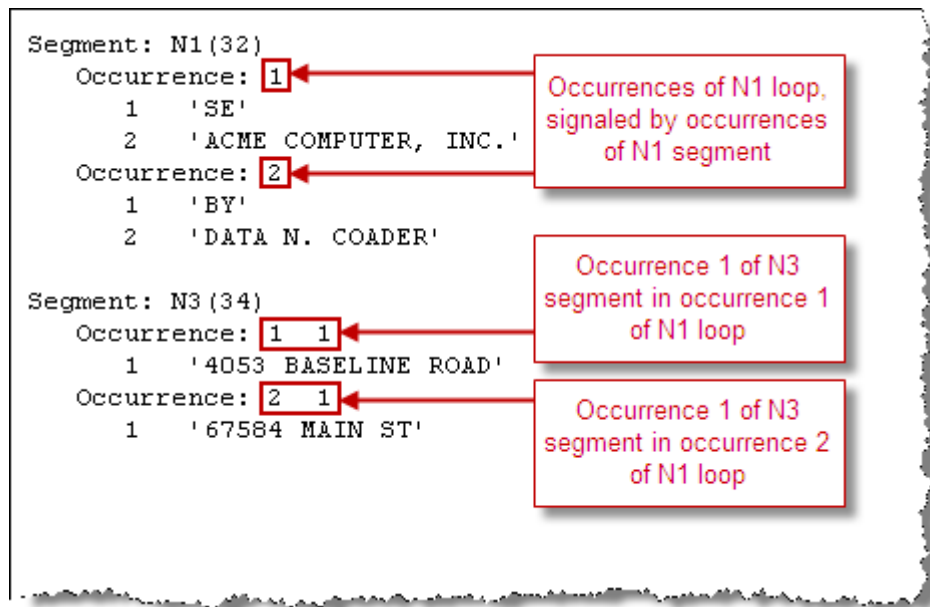
When there are repeating segments, each occurrence has an occurrence number, as you can see in the following example. The segment N1 occurs twice in the input document, once with seller information and once with buyer information.

```
Segment: N1(32)
Occurrence: 1
  1 'SE'
  2 'ACME COMPUTER, INC.'
Occurrence: 2
  1 'BY'
  2 'DATA N. COADER'
```

Segment N3 is in the N1 loop. It occurs twice: once in the first occurrence of the N1 loop and once in the second occurrence of the N1 loop.

The context list is a string of numbers that shows the occurrence number for each hierarchical level within which the segment occurs. To understand to which loop the level numbers refer, you must know the structure of the document. For example, you must know that an N3 that follows an N1 occurs in the N1 loop.

Read the context list from right to left, backwards, to determine where the segment occurs in the document. In the following example of two N3 segments, the location of the first occurrence of the N3 segment would be read as "Occurrence 1 of segment N3, in occurrence 1 of loop N1." The location of the second N3 segment should be read as, "Occurrence 1 of segment N3, in occurrence 2 of loop N1."



Example of Document Status, Statistics and Errors on Processing Report

After all documents have been parsed, MW Translator displays status, statistics and error information for use in such things as acknowledgments.

The SSEDOC definition, which is in the **Documents** folder of the INTERNAL, 1 standard version, uses the SET segment. The SSEDOC may contain further information, not shown on the current report, about segments and elements that would typically contain error information used in acknowledgments.

Document: INTERNAL, 1, SSEDOC

Description: Internal SSEDOC data element store

	L/S	Level	Area	Seq	Tag	Description
▶	L	0	1	10	SET	Loop
	S	1	1	20	SET	Set Level Info
	S	1	1	30	SETER	Set Level Error Info
	S	1	1	40	STUSR	Set Level User Info
	L	1	1	50	SEG	Loop
	S	2	1	60	SEG	Segment Level Info
	S	2	1	70	SEGER	Segment Level Error Info

Double-click the SET segment to view the elements. You could also view the segment directly from the **Segments** folder.

Segment: INTERNAL, 1, SET

Description: Set Level Info

	Seq	Ele ID	Description
▶	1	SET01	Set ID
	2	SET02	Set Version Number
	3	SET03	Set Release Number
	4	SET04	Transaction Set Reference Number
	5	SET05	Segment Count
	6	SET06	Translation Status
	7	SET07	Controlling Agency
	8	SET08	Association

The following example shows information for the first and part of the second of three documents MW Translator has parsed. The context list here contains a reference to the functional group wrapper level in which it occurs, which is the first number. When you have multiple functional groups in an interchange, this first number would increment to represent the number of the functional group.

```

Dump of Data Element Store: DOCSSE

Segment: SET (1)
Occurrence: 1 1
  1 '850'
  2 '003030'
  4 '0001'
  5 '23'
  6 'A'
  7 'X'
Occurrence: 1 2
  1 '850'
  2 '003030'
  4 '0002'
  5 '23'
  6 'A'
  
```

Information for the first document:
Occurrence 1 of the loop SET in occurrence 1 of the functional group wrapper

Information for the second document:
Occurrence 2 of the loop SET in occurrence 1 of the functional group wrapper

Example of Wrapper Data on Processing Report

The processing report displays the data parsed from the input wrapper. In the following example, the input wrapper is an X12 ISA.

```
Segment: ISA(1)
Occurrence: 1
1 '00'
2 ' '
3 '00'
4 ' '
5 'ZZ'
6 'ICH-SEND-ID '
7 'ZZ'
8 'ICH-REC-ID '
9 '940519'
10 '1018'
11 'U'
12 '00303'
13 '000010001'
14 '1'
15 'P'
16 ']'

Segment: GS(2)
Occurrence: 1 1
1 'PO'
2 'FG-SEND-ID'
3 'FG-REC-ID'
4 '940519'
5 '1018'
6 '100010001'
```

Occurrence 1 of wrapper segment ISA

Occurrence 1 of wrapper segment GS in occurrence 1 of loop ISA

Example of Wrapper Status, Statistics and Errors on Processing Report

After all wrapper segments have been parsed, MW Translator displays status, statistics and error information for use in such things as acknowledgments.

The SSEWRAP definition, which is in the **Documents** folder of the INTERNAL, 1 standard version, uses the WRAP segment. The SSEDOC may contain further information, not shown on the current report, about segments and elements that would typically contain error information used in acknowledgments.

Document: INTERNAL, 1, SSEWRAP

Description: Internal SSEWRAP data element store

	L/S	Level	Area	Seq	Tag	Description
▶	S	0	1	10	WRAP1	Interchange (Level 1) Info
	S	0	1	20	WERR1	Wrapper (Level 1) Error Info
	S	0	1	30	WUSR1	Wrapper (Level 1) User Info
	L	0	1	40	WR1	Loop
	S	1	1	50	WRAP2	Functional Group (Level 2) Info
	S	1	1	60	WERR2	Functional Group (Level 2) Error Info
	S	1	1	70	WUSR2	Functional Group (Level 2) User Info

Double-click the WRAP1 segment to view the elements. You could also view the segment directly from the **Segments** folder.

Segment: INTERNAL, 1, WRAP1

Description: Interchange (Level 1) Info

	Seq	Ele ID	Description
▶	1	WR01	Control Reference
	2	WR02	Status
	3	WR03	Count of the Contents
	4	WR04	Accepted Count
	5	WR05	Rejected Count
	6	WR06	Wrapper Id Code
	7	WR07	Wrapper Set Id

The following example shows information for the two levels of wrappers MW Translator has parsed: the interchange-level wrapper, and the functional group level wrapper.

```
Dump of Data Element Store: WRAPSSE

Segment: WRAP1 (1)
  1  '000010001'
  2  'A'
  3  '1'
  4  '1'
  5  '0'
  8  '00303'
 10  'ICH-SEND-ID'
 11  'ZZ'
 13  'ICH-REC-ID'
 14  'ZZ'
 19  '          '
 20  '00'
 21  'P'

Segment: WRAP2 (4)
Occurrence: 1 1
  1  '100010001'
  2  'A'
  3  '3'
  4  '3'
  5  '0'
  8  '00303'
 10  'FG-SEND-ID'
 13  'FG-REC-ID'
 16  'PO'
```

Information for the first-level wrapper, ISA

Information for the second-level wrapper, GS
Occurrence 1 of the GS wrapper in occurrence 1 of the ISA

Generate Errors

There are two types of generate errors:

- Errors from the Workbench when you generate text files
- Errors from the TRM when it generates output data

Here is how MW Translator handles some basic exceptions during generation:

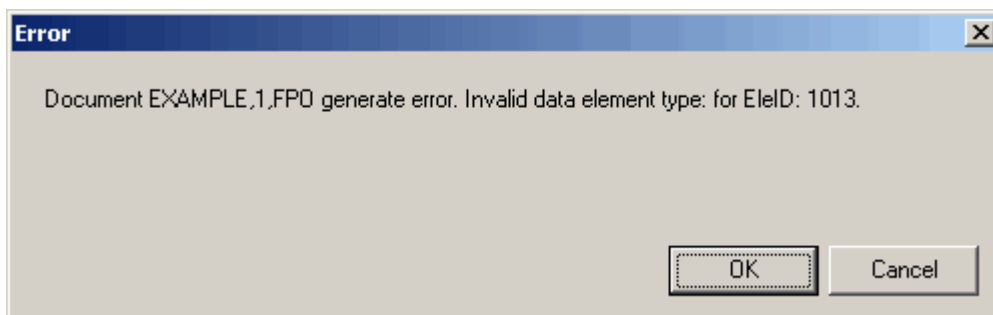
Location of Exception	Behavior
Document	Attempts to roll back processing prior to the parsing of the input document and skips the entity to continue with the next one. Accept or reject the document, depending on configurations, and continue with the next document in the interchange. Control reference counters are reset.
Wrapper header	Attempts to roll back processing prior to the parsing of the wrapper and skips the entity to continue with the next one. It progresses from the most specific level, document, to the least specific, interchange. That level will be accepted or rejected, depending on configurations. Any output or acknowledgments generated to this point specific to this level will be aborted.
Wrapper trailer	Entire inbound interchange will be rejected.

IMPORTANT: If a reject occurs during generation of an acknowledgment, then processing is aborted.

Generating Configurations

The Workbench allows you to partially define entities to provide the greatest degree of freedom in doing so. This also provides the opportunity to go to the next stage of generation with partially complete definitions. The generate process checks for such problems.

For example, the generate process will display an error if you partially define an element. The following figure shows an example of such an error during generate.

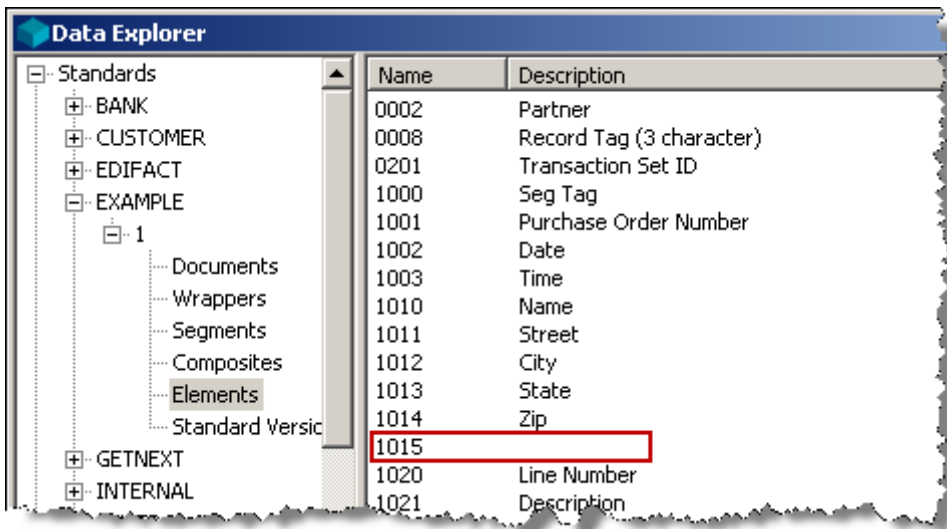


Troubleshooting Technique

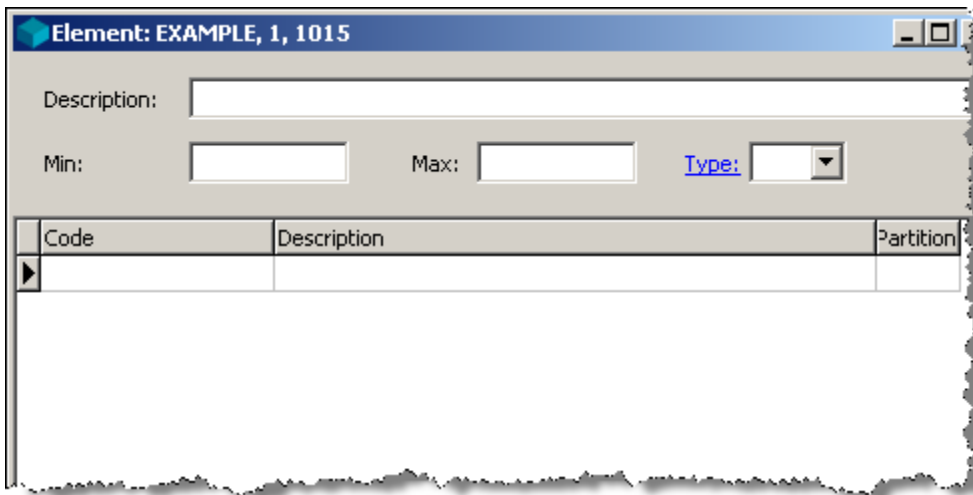
The Workbench checks for consistency in your configurations when you generate your text files. If it finds any problems, it displays an error message. If you see such a message, check the text file for error flags. When you fix the error, regenerate the text file.

- Always check Data Explorer or Partner Explorer for visible cues of missing or wrong information.
- From the **View** menu, select **Refresh** or press **F5** to make sure you are looking at the most recent information. This is particularly important in a multi-user environment.

In this example, notice the missing description for element 1013 in the right pane of Data Explorer, as displayed in the following figure.

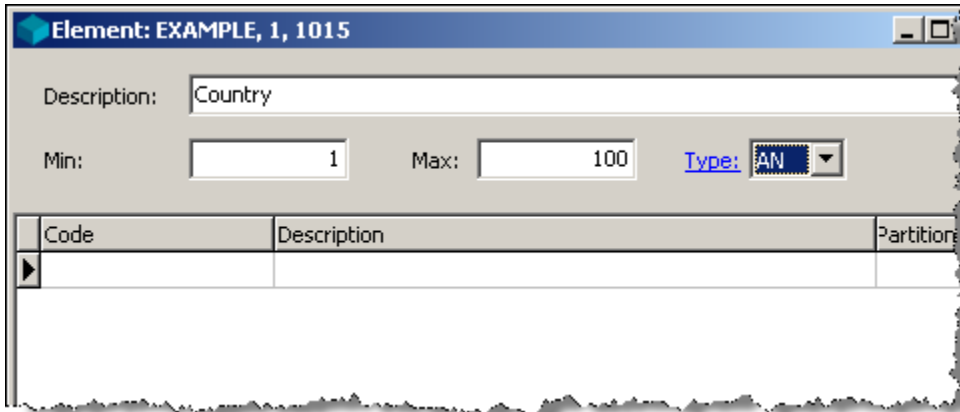


- Open the suspect definition for more clues. When you look at the definition for this element, you will see that it is not completely defined, as displayed in the following figure.



Solution

- 1 Enter the missing information, which in this example is the element definition, as shown below:



- 2 Regenerate the document.

Generating Data

MW Translator generates segments as it finishes parsing certain parts of the input message. The process varies depending on the structure of the output. To follow the process MW Translator uses, select any of the **Print Processing Trace** options on the Modify Options window. This information shows the output data that was generated.

This is an example of generate comments that appear on the report.

```

...Output 1: Checking map for segment '***'...
...Generating output 1, segment '***' (1)...
...Generating output 1, element seq 1...   Element value: "***"
...Generating output 1, element seq 2...   Element value: "TESTSEND"
...Generating output 1, element seq 3...   Element value: "FPO"
Map Name: EXAMPLE-X850

...Generating output 1 document: EXAMPLE, 1, FPO with map: EXAMPLE-X850...
...Output 1: Checking map for segment 'HDR'...
...Generating output 1, segment 'HDR' (1)...
...Generating output 1, element seq 1...   Element value: "HDR"
...Generating output 1, element seq 2...   Element value: "PO12345"
...Generating output 1, element seq 3...   Element value: "051994"
...Generating output 1, element seq 4...   Element value: "1018"
...Generating output 1, element seq 5...   Element value: ""
...Output 1: Checking map for segment 'BUY'...
...Generating output 1, segment 'BUY' (2)...
...Generating output 1, element seq 1...   Element value: "BUY"
...Generating output 1, element seq 2...   Element value: "DATA AL CODE"
    
```

NOTE: When the TRM does contents validation only or translation, it routes the entire interchange. If the **Error Action** is set to **Reject** (Trade Agreement Profile window, **Options** tab) and there is an error in the document, it deletes rejected documents from the outgoing interchange, adjusting trailer counts as necessary. When there are errors in the interchange or functional group level wrappers, the TRM rejects the entire interchange.

General Reference

Title Bar

You may use multiple databases for testing. Your current database environment selection displays on the title bar. The following title bar shows a selected database environment.

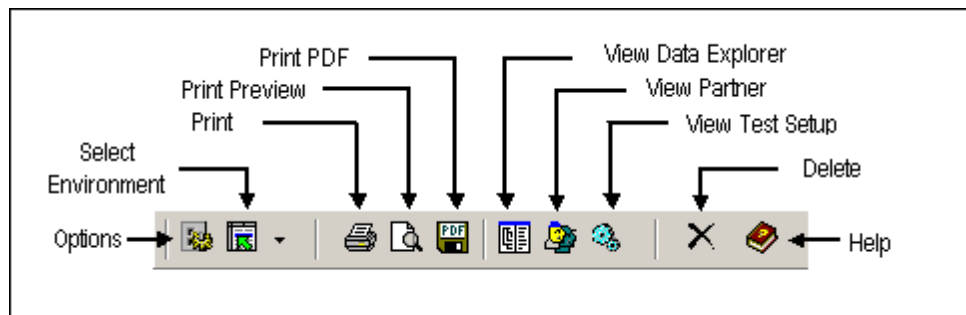


To select a different database environment, use the **Select Environment** button to display the list.



Toolbar

The following figure shows you the default toolbar buttons. Others appear when certain entities are selected.


















Menus and Task Icons

The following tables describe the menu structure for the Workbench with an explanation of what the commands do and any associated icons for specific tasks that you might be able to choose from the toolbar.





File Menu (Workbench)

The following commands appear on the **File** menu.

Command	Shortcut	Icon	Description
Add			Adds a new entity in selected folder
Save	Ctrl+S		Saves selected configuration
Close			Closes current window
Select Environment			Presents list of database environments from which to select. Great for testing.
Options			Displays Modify Options window to change Workbench environment configurations.
Import			Imports definitions to current database from selected file that was previously exported. Useful for moving information between PCs.
Export			Exports definitions to transfer file from database. Useful for moving information between PCs.
Copy to			Copies definitions from one database to another.
Delete			Delete the selected entity.
Rename			Renames selected entities.
Properties			Presents properties page of selected entity.
Print	Ctrl+P		Prints selected definitions to a printer.
Print Preview			Displays printed report of selected definition on console.
Print PDF			Prints selected definitions to a PDF file.
Exit			Exits the Workbench.




Edit Menu (Workbench)

The following commands appear on the **Edit** menu. Many of these commands only appear for the Edibasic Edit window.

Command	Shortcut	Icon	Description
Cut	Ctrl+X		Cuts selected text from Edibasic Edit window.
Copy	Ctrl+C		Copies selected text from Edibasic Edit window to clipboard.
Paste	Ctrl+V		Pastes contents of clipboard to Edibasic Edit window.
Delete	Ctrl+Del		Delete text selected on Edibasic Edit window.
Find	Ctrl+F		Locates specified text on current page
Search again	F3		Locates next instance of specified text on current page.
Select All	Ctrl+A		Selects all items.

View Menu (Workbench)

The following commands appear on the **View** menu.

Command	Shortcut	Icon	Description
Data Explorer			Displays Data Explorer window.
Partner Explorer			Displays Partner Explorer window
View Test Setup			Displays Test window.
Next	F8		Displays next entity configuration in database.
Prior	F7		Displays prior entity configuration in database
Refresh	F5		Re-displays contents of selected window.


Generate Menu (Workbench)

The following commands appear on the **Generate** menu.

Command	Shortcut	Icon	Description
StdID File			Generates text file from all standard ID database records
Partner File			Generates text file from all Partner and Partner relationship database records.
Profile File			Generates text file from all trade agreement and acknowledgment profile database records.
Xref File			Generates text file from all cross-reference database files.
All			Generates all definitions for all entities in database (allows you to specify wrappers, documents, and maps separately).
Current 10 configurations			Generates selected configurations from list of 10 most recently accessed and currently active windows.




Tools Menu (Workbench)

The following commands appear on the **Tools** menu.

Command	Shortcut	Icon	Description
Pack database			Reclaims space of deleted records for entire database.
Partner Wizard			Activates Partner Wizard to help you create partnerships.
Load Standards			Allows you to select an entire public standard to load to your current database.

Check Errors Menu (Workbench)

The following commands appear on the **Check Errors** menu. Note that this menu appears only if you are working in an Edibasic Edit window.

Command	Shortcut	Icon	Description
Check Errors			Check the Edibasic syntax on the current page for errors and positions cursor to first error.
Next Error			Positions cursor to the next error.
Prior Error			Positions cursor to the prior error.


Windows Menu (Workbench)

The following commands appear on the **Windows** menu.

Command	Shortcut	Icon	Description
Arrange All			Arranges all open windows in cascading format.
Close All			Closes all open configuration windows, not including the explorer windows.
Minimize All			Minimizes all open windows, including the explorer windows.
Current windows			Lists all configuration windows that you currently have open, excluding the explorer windows.

Help Menu (Workbench)

The following commands appear on the **Help** menu.

Command	Shortcut	Icon	Description
Contents			Displays table of contents for help file.
Topic Search			Displays the index for help file.
About MW Translator			Displays the copyright and version information for the Workbench.

Command	Shortcut	Icon	Description
Workbench			

Copy Command

You use the **Copy to** command to move selected configuration definitions within and between databases.

IMPORTANT: When copying or importing Standard ID definitions from one MW Translator format to another MW Translator format, the copy utility will avoid creating unnecessary duplicates. This is not true when you copy or import definitions from Edikit versions prior to Edikit 4.0, in which case you may end up with some duplicate standard IDs. When you copy definitions from versions prior to Edikit 4.0, you must manually delete unnecessary duplicate Standard ID wrappers.

The following table indicates the definitions that users may copy in the **Copy Type** column, which are selectable from the right panes of the Data Explorer and Partner Explorer windows. Optionally, you may include definitions related to the entity being copied, which are listed in the **Related Definitions** column.

Copy Type	Related Definitions
Standard	Its documents, wrappers, segments, composites, elements, and the standard types for the standard version
Document	Its segments, composites and elements and the standard types for the standard version
Wrapper	Its segments, composites and elements and the standard types for the standard version
Segment	Its composites and elements and the standard types for the standard version
Composite	Its elements and the standard types for the standard version
Element	Standard types for the standard version
Map	Source document and wrapper and their related definitions; destination document or wrapper and its related definitions
Trade Agreement Profile	Input document and its related definitions; document map and its related definitions; wrapper map and its related definitions; summary map and its related definitions
Acknowledgment Profile	Source wrapper and its related definitions; acknowledgment map and its related definitions; acknowledgment wrapper map and its related definitions; summary map and its related definitions
Cross Reference	No related definitions

Copy Type	Related Definitions
Standard ID	Source wrapper and its related definitions; all listed trade agreement profiles and their related definitions; all listed acknowledgment profiles and their related definitions; default sending partner; default recipient partner
Partner	All listed trade agreement profiles and their related definitions; all listed acknowledgment profiles and their related definitions
Partner Relationship	Sending and recipient partners and their related definitions; all listed trade agreement profiles and their related definitions; all listed acknowledgment profiles and their related definitions

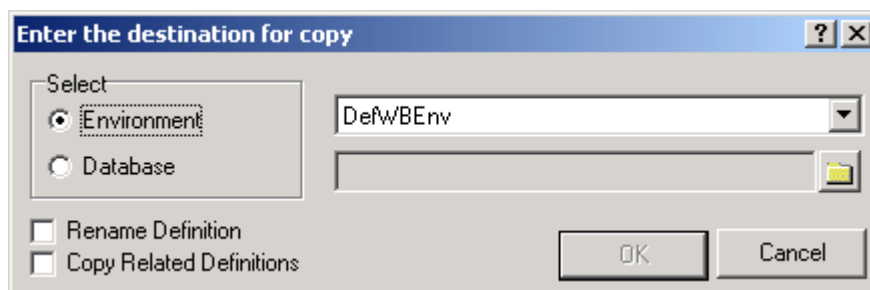
NOTE: You cannot copy input locations, and you cannot copy groups, because group definitions are part of partner and standard ID definitions.

You may copy definitions to another database, renaming the definitions or not. You may also copy definitions to the same database. For Data Explorer, when similar definitions already exist in the destination database and you do not initially choose to rename them, the Workbench will prompt you before it will overlay them. For Partner Explorer, you are not able to rename definitions during a **Copy** command, because these definitions are shared between the Workbench and the Operator Program.

When you select the **Copy to** command, the **Enter the destination for copy** dialog box appears.

Enter the Destination for Copy Dialog Box

You may copy your selected definitions to your current database or to another database by specifying the database environment or the database path. Either select the database environment from the list, or use the **Browse** button to locate the database.



Select Environment or Database

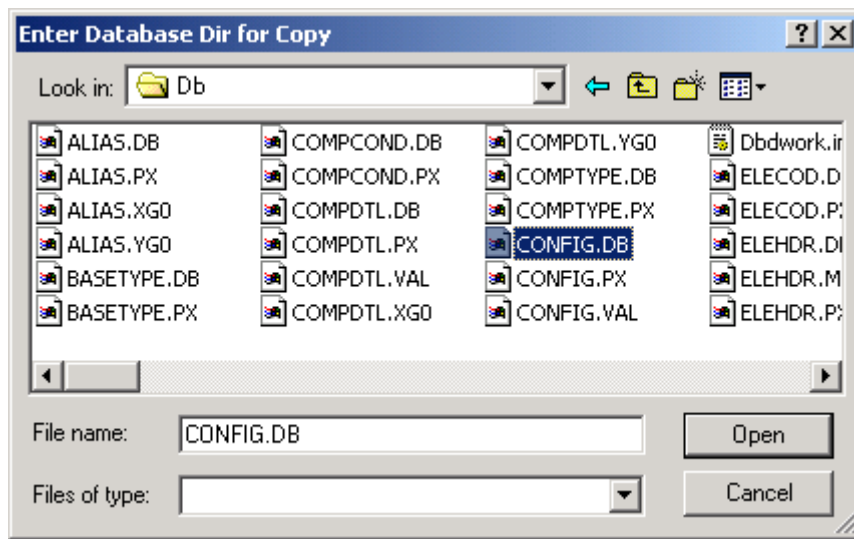
Select where you want to copy your definitions. Select **Environment** when the environment is on the environment list. Select **Database** when it is not on the list. When you copy to your current environment, you must also rename the definitions.

Environment Value

From the drop-down list, select the database environment to which you want to copy your definitions. When you select your current environment, you must also check the **Rename Definition** box.

Database Value

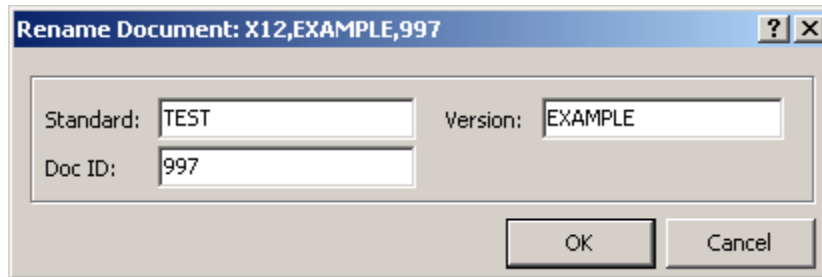
Locate or enter the path of the database to which you want to copy your definitions. It may be another database or your current database. When you select your current database, you must also check the **Rename Definition** box. When the **Enter Database Dir for Copy** dialog box appears, the **CONFIG.DB** file is selected by default. You may also decide to copy related definitions, those that are referenced in the definition you want to copy.



Enter Database Dir for Copy Dialog Box

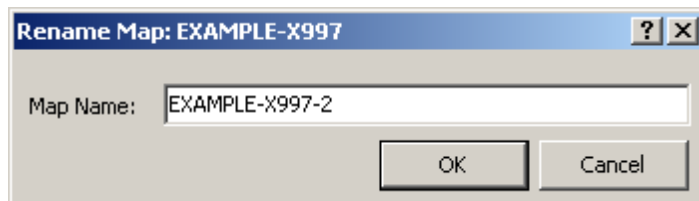
Rename Definition

Select this box to rename the definitions that you want to copy. You must select this box when you are copying to and from the same environment or database. Note that during a copy, you cannot rename standard identification, partner or partner relationship definitions. When the **Rename** dialog box appears, enter the information requested to uniquely identify the entity in the target database.



The screenshot shows a dialog box titled "Rename Document: X12,EXAMPLE,997". It has three input fields: "Standard" containing "TEST", "Version" containing "EXAMPLE", and "Doc ID" containing "997". At the bottom right, there are "OK" and "Cancel" buttons.

Rename Dialog Box (Document)



The screenshot shows a dialog box titled "Rename Map: EXAMPLE-X997". It has one input field labeled "Map Name" containing "EXAMPLE-X997-2". At the bottom right, there are "OK" and "Cancel" buttons.

Rename Dialog Box (Map)

Copy Related Definitions

Select this box to copy related definitions. For example, if you want to copy a partner relationship, this selection will also copy the definitions for the sending and recipient partners. If you want to copy the definitions for a Standard ID definition in a location, this will copy the location and any associated trade agreements, documents and maps.

Procedures (Copy Command)

This procedure copies a definition within the same or between different databases. You can also choose to copy related definitions. You select the destination database by selecting an environment or a **config.db** file.

NOTE: You cannot copy source locations, and you cannot rename standard ID, partner or partner relationship definitions.

To Copy a Definition

- 1 In the left pane, expand your definitions until the one you want to copy appears in the right pane.
- 2 In the right pane, select the definition you want to copy.
- 3 From the **File** menu, select **Copy to**.

The **Enter the Destination for Copy** dialog box appears.

- 4 Specify the database or the environment as the target:

- To specify the environment:
 - Select the **Environment** button from the Select box.
 - Select the appropriate environment from the list.

– or –

- To specify the database rather than the environment:
 - Select **Database** from the **Select** box, and then the **Browse** button next to the data entry box.
 - The **Enter Database Dir for Copy** dialog box appears.
 - Select the location of the database to which you want to copy the definition. The partial path of the database file should be **\CFG\CONFIG.DB**.
 - Select **Open**.
 - The **Enter the destination for copy** dialog box appears with the database path.

- 5 For definitions within Data Explorer, to change the name of the definition:

- a) Check the **Rename Definition** box.
- b) When the **Rename** dialog box appears, type the new name and select **OK**.

NOTE: It is not possible to rename definitions within Partner Explorer, because both the Workbench and the Operator Program can change these definitions.

- 6 To copy other definitions that are referenced by the definition you are copying, select the **Copy Related Definitions** check box.
- 7 Select **OK**.

You will receive a confirmation for successful completion of your copy request.

Export Command

You use the **Export** command to export configuration definitions to a new or existing flat file. If the file already exists, the Workbench allows you to make a choice to append the information to the file or replace it. This feature is useful as a quick and selective backup or to transport the definitions to another system.

You may export definitions from the Test window after a translation, from Data Explorer, or from Partner Explorer. The **Export** command always exports related definitions. For example, if you export definitions for a translation from the Test window, the export file will contain various types of definitions, including input and output standards, maps, standard ID, a trade agreement, and perhaps acknowledgments and partners.

The following table indicates the definitions that will be exported, either from Test or from the right panes of the Data Explorer and Partner Explorer windows.

Export Type	Exportable Definitions
Test	All definitions required to run the test
Standard	The standard version and its documents, wrappers, segments, composites, elements, and the standard types for the standard version
Document	Document with its segments, composites and elements and the standard types for the standard version
Wrapper	Wrapper with its segments, composites and elements and the standard types for the standard version
Segment	Segment with its composites and elements and the standard types for the standard version
Composite	Composite with its elements and the standard types for the standard version
Element	Element and the standard types for the standard version
Map	Source document and wrapper and their related definitions; destination document or wrapper and its related definitions
Trade Agreement Profile	Input document and its related definitions; document map and its related definitions; wrapper map and its related definitions; summary map and its related definitions
Acknowledgment Profile	Source wrapper and its related definitions; acknowledgment map and its related definitions; acknowledgment wrapper map and its related definitions; summary map and its related definitions
Cross Reference	Cross-reference table

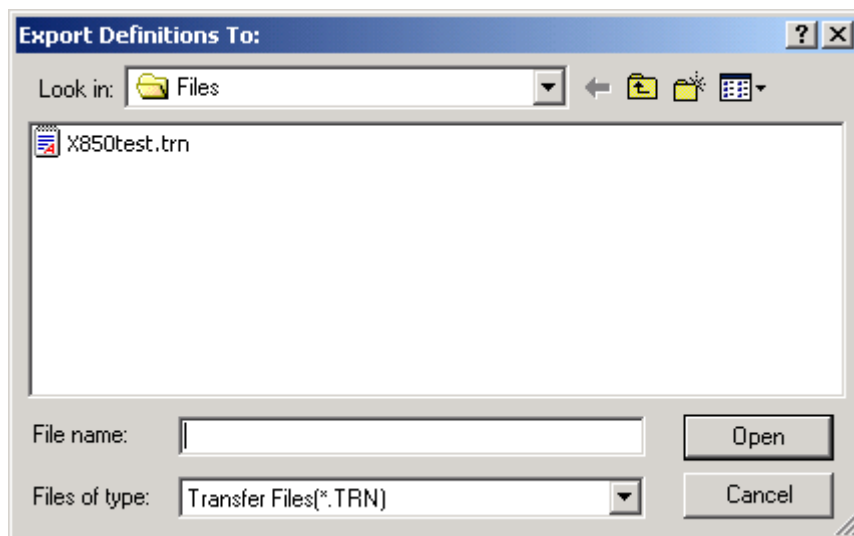
Export Type	Exportable Definitions
Standard ID	Input location; source wrapper and its related definitions; all listed trade agreement profiles and their related definitions; all listed acknowledgment profiles and their related definitions; default sending partner; default recipient partner
Partner	Partner and all listed trade agreement profiles and their related definitions; all listed acknowledgment profiles and their related definitions
Partner Relationship	Sending and recipient partners and their related definitions; all listed trade agreement profiles and their related definitions; all listed acknowledgment profiles and their related definitions

NOTE: You cannot export groups, because group definitions are part of partner and standard ID definitions. You cannot export an input location.

When you select the **Export** command from the **File** menu or the **Export Translation Definitions** button



from the toolbar for the Test window, the **Export Definitions To** dialog box appears.



Export Definitions To Dialog Box

Procedures (Export Command)

This procedure exports a definition to a flat file. This command also exports related definitions.

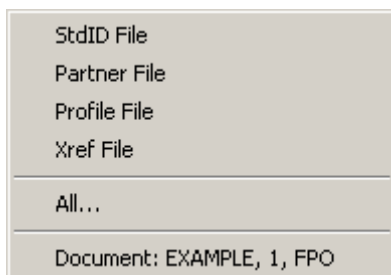
To Export Definitions to Transfer Files

This procedure exports a selected definition and any related definitions.

- 1 In the left pane of the appropriate window, expand your definitions until the one you want to export appears in the right pane.
- 2 In the right pane, select the definition you want to export.
- 3 From the **File** menu, select **Export**.
The **Export Definitions To** dialog box appears.
- 4 Specify a location for the file and a file name.
An extension of **.TRN** is added as required.
- 5 Select the **Open** button to begin the export process.
- 6 When the file already exists, a dialog box appears to allow you to append the information to the file or overwrite it.
An information box appears indicating progress.
- 7 To interrupt the export process, select the **Cancel** button.

Generate Command

When testing, the TRM uses definitions from generated text files. From the **Generate** menu, you select which set of definitions you want to generate as text files: standard IDs, partners, trade agreement and acknowledgment profiles, and cross-references. You can also select to generate them all or one of the ones listed as an open window.

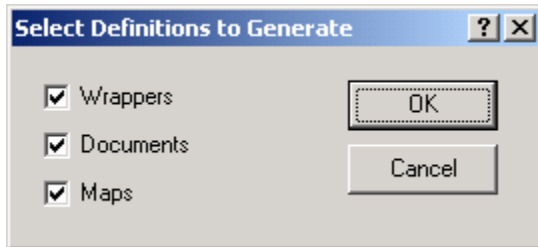


Depending on your selection, the Workbench generates text files that contain the associated entities listed in the following table.

Generate Option	Definitions Included
StdID File	Standard Identifications
Partner File	All configured partners and partner relationships

Generate Option	Definitions Included
Profile File	All configured trade agreements and acknowledgments
Xref File	All configured cross-references
All	All files, with an option to separately select wrappers, documents or maps
Listed entity (wrapper, document, map)	Definition for the listed window that you select from the Generate menu.

When you select **All**, the **Select Definitions to Generate** dialog box appears. To include wrappers, documents, or maps in the generated text files, check the appropriate boxes.



Select Definitions to Generate

Procedures (Generate Command)

The following procedures describe various methods of generating text files for testing.

To Generate Specific Types of Definitions or All Definitions

- 1 To generate all definitions:
 - a) From the **Generate** menu, select **All**.
A **Select Definitions to Generate** dialog box appears.
 - b) Check the boxes of those entities you wish to optionally generate: wrappers, documents or maps.
A **Generating Definitions** information box appears to indicate progress.

– or –
- 2 To generate specific types of definitions:
 - a) From the **Generate** menu, select the type of definitions you want to generate: standard ID file, partner file, profile file or xref file.
A **Generating Definitions** information box appears to indicate progress.

To Generate a Specific Definition

Use this procedure to transfer one configuration definition to a text file.

- 1 You can generate files using one of two methods, depending on whether the window is open and selected:
 - a) In the left pane, expand the folder that has the definition you want to generate, and in the right pane, select the desired definition.
 - or –
 - b) If you have window open already, select the window. The program will generate whatever currently has focus.
- 2 From the **Generate** menu, select the name of the entity at the bottom of the list.
A **Generating Definitions** information box appears to indicate progress.

Import Command

You can import configuration definitions from flat files. Typically, these files were created previously using the **Export** command. The **Export** command always exports related definitions. For example, if you exported definitions for a translation from the Test window, the export file will contain various types of definitions, including input and output standards, maps, standard ID, a trade agreement, and perhaps acknowledgments and partners. When you use the **Import** command to import these definitions, you will be able to select some or all of the definitions. You will also be able to rename or overlay the definitions, in case they already exist in the database.

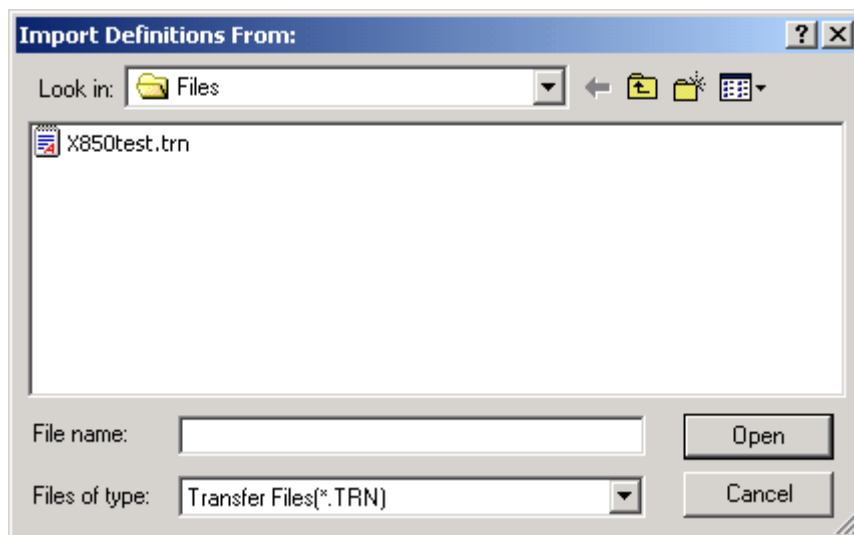
The following table shows the type of definitions that may be imported from the file in the **Import Type** column. These definitions will include other definitions integral to the entity being copied, which are listed in the **Related Definitions** column.

Import Type	Related Definitions
Standard	Its documents, wrappers, segments, composites, elements, and the standard types for the standard version
Document	Its segments, composites and elements and the standard types for the standard version
Wrapper	Its segments, composites and elements and the standard types for the standard version
Segment	Its composites and elements and the standard types for the standard version
Composite	Its elements and the standard types for the standard version

Import Type	Related Definitions
Element	Standard types for the standard version
Map	No related definitions
Trade Agreement Profile	No related definitions
Acknowledgment Profile	No related definitions
Cross Reference	No related definitions
Standard ID	No related definitions
Partner	No related definitions
Partner Relationship	Sending and recipient partners and their related definitions

NOTE: Use the **Load Standards** command to import standards, which imports the entire standard.

When you select the **Import** command, the **Import Definitions From** dialog box appears.

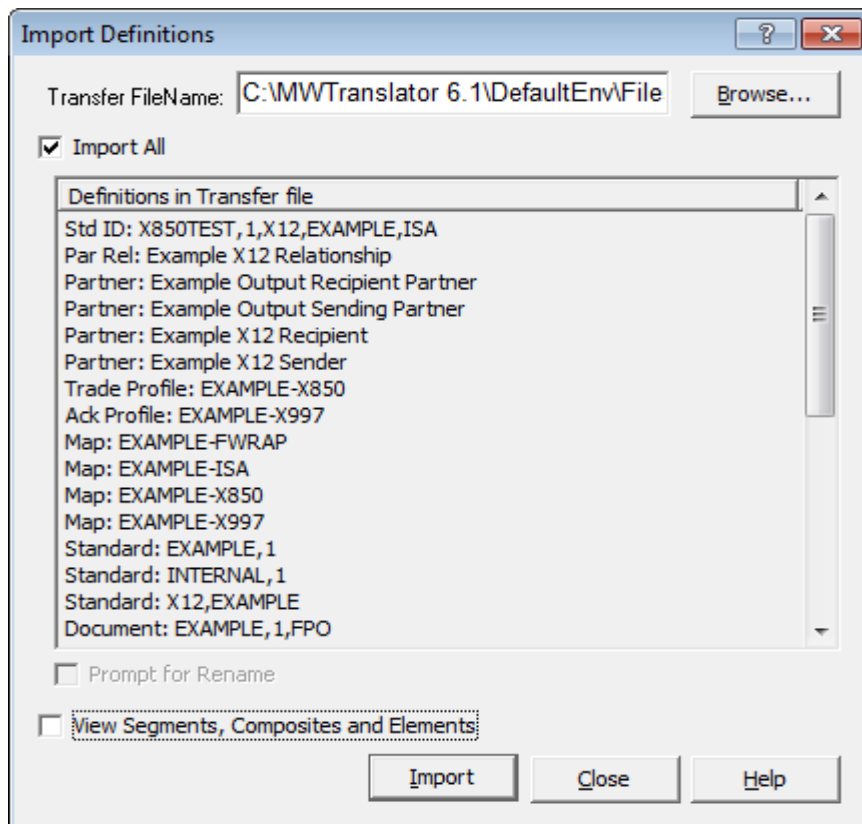


Import Definitions From Dialog Box

Import Definitions Window

You may import definitions to the current environment with or without renaming them. The default action is to import all definitions listed, which will prompt you to replace definitions as required. Alternatively, you can select the definitions to import and optionally rename them. You may also view all segments, composites and elements and import those individually.

NOTE: Only the first definition within the Import Definitions window can be renamed. If you need to rename a definition that is not the first one, you will need to export only the definition you want to rename and import it using the rename option.



Import Definitions Window

Transfer FileName (Import Definitions)

This field displays the name of the transfer file that you selected from the initial dialog box and from which you want to import database definitions. You may also use the **Browse** button to select a file, typically from a **Files** subdirectory. The typical extension for a transfer file is .trn.

Import All

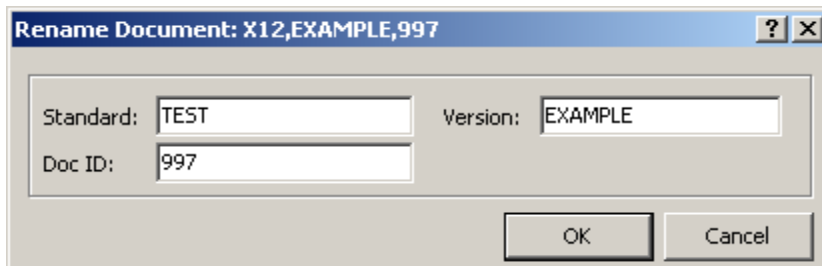
Check the **Import All** box to import all definitions listed in the list box. If the definitions already exist in the target database, you will be prompted to overwrite them. To rename the first definition or to select only certain definitions from the list to import, uncheck **Import All**. To rename the first definition displayed, check the **Rename** box.

Definitions in Transfer File

Once you have chosen a transfer file name, this lists the definitions that the file contains. You may optionally display the segments, composites and elements. By default, all definitions are ready for import. You can scroll through the list using the scroll bars, and select or deselect individual definitions. Selected definitions will be highlighted in a different color. To rename the first definition, you must uncheck the **Import All** box. After you select the first definition displayed, you may rename it by checking the **Prompt for Rename** box.

NOTE: Only the first definition within the Import Definitions window can be renamed. If you need to rename a definition that is not the first one, you will need to export only the definition you want to rename and import it using the rename option.

When you check a box and select **Import**, a rename dialog box appears. Enter the information requested to uniquely identify the entity in the target database.



Rename Dialog Box (Import)

Prompt for Rename

To rename the first definition, uncheck **Import All**, select the first definition displayed, check the **Prompt for Rename** box and then click the **Import** button.

View Segments, Composites and Elements

Select this box to show all definitions, including segments, composites and elements associated with all the higher lever definitions. Use this view to import individual segments, composites and elements.

Import Button

Select the **Import** button to complete your request.

Procedures (Import Definitions)

This procedure imports a definition from a transfer file to a destination database. In order to use this procedure, you must have already created a transfer file using the **Export** command.

To Import Definitions to a Database

All definitions are imported to the current environment. If you want to change the database to which you will import the definitions, you must do so before you begin this process.

- To switch to another environment, refer to the procedure *To Switch between Database Environments* (on page 529).
- To switch to another database, but not the entire environment, refer to the procedure *To Change Database, Configuration, Input, Output, Report or Transfer File Locations for the Workbench* (on page 519).

1 From the **File** menu, select **Import**.

The **Import Definitions From** dialog box appears.

2 Select the appropriate name of the transfer file.

The **Import Definitions** dialog box appears with the file name in the **Transfer FileName** box.

3 Determine whether you want to import all of the definitions or only specific definitions:

- To import all definitions, the **Import All** button should be checked.

By default, all definitions are ready to be imported.

– or –

- To import some of the definitions or the first definition displayed in order to rename it, uncheck the **Import All** box and select the definition(s).

- To select multiple definitions, use the **SHIFT** or **CTL** keys.

- To change the name of the first definition displayed, check the **Prompt for Rename** box.

When the first definition displayed is selected, a **Rename** dialog box appears. Enter the information requested to uniquely identify the entity in the target database. Select the **Import** button.

4 If you attempt to import a definition that exists within the destination database, a confirmation box appears at the first duplicate entity. You must select **Yes** (overlay this entity), **No** (do not overlay this entity), **YesToAll**, (yes, and do not ask again), or **NoToAll** (no, and do not ask again).

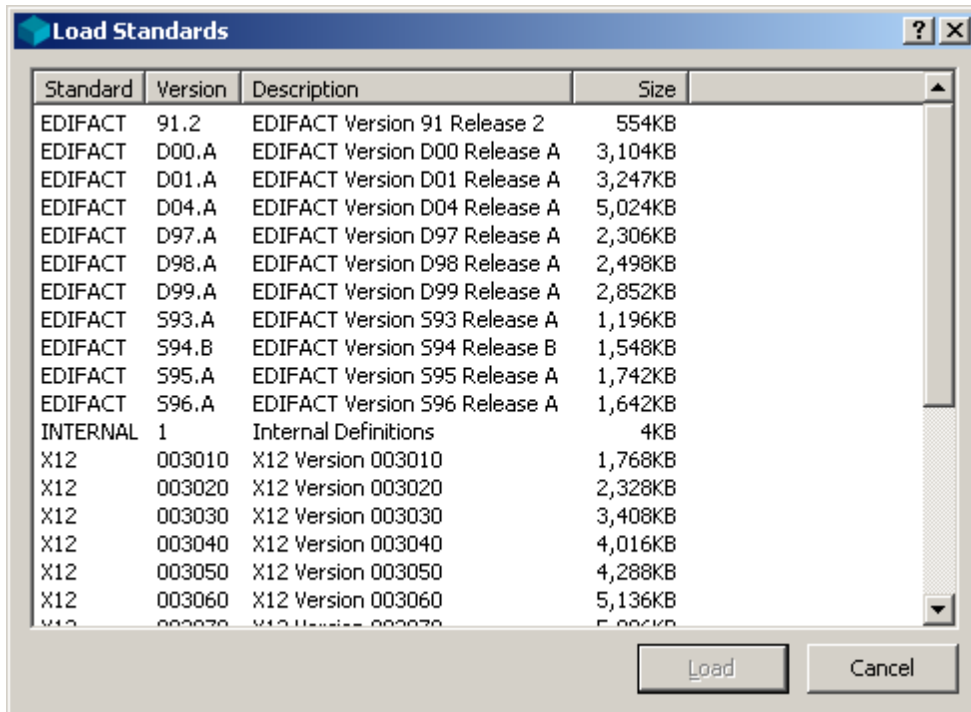
NOTE: Only the first definition within the Import Definitions window can be renamed. If you need to rename a definition that is not the first one, you will need to export only the definition you want to rename and import it using the rename option.

NOTE: To select individual segments, composites or elements, make sure you check the **View segments, composites and elements** box.

Load Standards Command

You use the Load Standards window to load entire definitions for a standard from a disk file. These files are stored in the Standards directory when you install the Workbench. You cannot rename the standards using this window. If you want to install specific definitions for a standard or you want to rename the standard, you should use the Import command.

When you select **Load Standards** from the **Tools** menu, the **Load Standards** dialog box appears. The list box displays all standards that were installed in the **Standards** directory. You select one or more standards that you want to load. The Workbench loads all definitions associated with this standard.



Load Standards Dialog Box

Procedures (Load Standards Command)

You can load an entire standard to a database using the Load Standards command or the Import command. If you want to load only part of the standards definitions or rename any of the definitions, refer to *Procedures (Import Definitions)* (on page 505). This procedure loads definitions to your current database. If you want to change only the database location, you must first do so from the Directories page in the Modify Options window.

To Load an Entire Standard to Your Database

- 1 From the **Tools** menu, select **Load Standards**.
- 2 From the **Load Standards** dialog box, select one or several of the standards that you want to load to your current database.
- 3 Select **OK** to complete the process. An information box appears indicating progress.
- 4 If you want to interrupt the process, select the **Cancel** button.

Options Command

The **Options** command displays the Modify Options window. Here users may change parameters that affect the current environment, including:

- Report options
- Report exceptions limit
- Hypertext color
- Locations of root and specific directories
- User exit DLLs

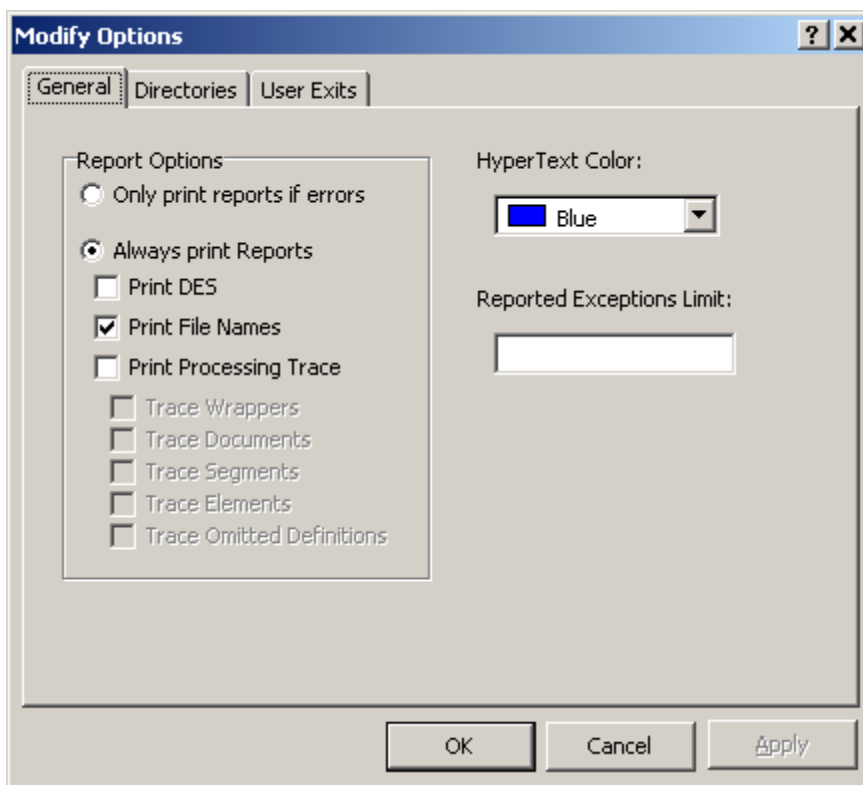
During installation, these values are set according to the install parameters, and appropriate target directories are entered.

You access this window, select the **Options** button  from the toolbar or **Options** from the **File** menu.

(Modify Options) General Page

From the **General** tab of the **Modify Options** dialog box, you control the following:

- When to print a processing report
- What information to print on the report, including trace information for debugging
- The color of text on configuration windows to identify text on which you can click to jump to other windows
- The maximum number of exceptions reported on the processing report



General Page (Modify Options Dialog Box)

Report Options

These selections affect whether you suppress or generate a processing report. You can print information about the status of the documents and wrappers, the contents of the data element store, and the file names of the configuration files used during processing. You can also trace what MW Translator is doing during processing, which is a powerful debugging tool.

Only Print Reports If Errors (Report Options)

Only Print Reports If Errors prints a processing report when there are errors or print statements in a map. The processing report includes the status of all wrappers and documents, as well as errors and the results of any print statements used for debugging.

Always Print Reports (Report Options)

Always Print Reports prints both the configuration report and the processing report. The processing report includes the status of all wrappers and documents, as well as errors and the results of any print statements used for debugging. For more information, please refer to the topic, *Runtime Exceptions* (on page 423).

The statuses that will print on the report are as follows:

Status Value	Description
A	Accept, with or without errors
R	Reject, continue processing if possible, otherwise abort
S	Security reject, continue processing

Print DES (Report Options)

Print DES prints the location and contents of the information stored in memory in the data element store (DES), which contains the input document, wrappers, and status, statistics and errors information. To print the data after it is parsed but before it is stored internally in the DES, or after it is generated, select the option, **Trace Elements**.

Print File Names (Report Options)

Print File Names prints the names of the text files it is looking for or has used to parse and generate document. You typically use this command to debug maps.

Print Processing Trace (Report Options)

Print Processing Trace is a powerful debugging tool. It allows users to view what the TRM is doing as it processes the input and creates the output. If you do not select any options below this level, the report contains the same information as produced with the **Always Print Reports** option.

Trace Wrappers (Report Options)

Trace Wrappers prints additional information for the process MW Translator follows when parsing and generating wrappers, including the following:

- Definitions it loads into memory to parse the input wrapper
- Level of wrapper it expects to parse next
- Level of wrapper found or did not find
- Acknowledgment definitions it is checking
- Level of wrapper it expects to generate next
- Level of wrapper generated or not generated

NOTE: For more information about the content of the wrappers, also check the **Trace Segments** and **Trace Elements** options.

Trace Documents (Report Options)

Trace Documents prints the information for the process MW Translator follows when parsing and generating documents, including the following:

- When it expects to find a document in the parsed input data
- When a document definition is found or not found
- What partner information it uses to find trade agreement
- Definitions it loads into memory to parse the input document
- When it parses the document
- Definitions for output document and map that it loads into memory to generate output
- When it generates the output

NOTE: For more information about the content of the documents, also check the **Trace Segments** and **Trace Elements** options.

Trace Segments (Report Options)

Trace Segments prints the information about the process MW Translator follows when parsing and generating segments and loops for wrappers and documents. It shows those segments or loops that actually exist in the input and output including the following:

- List of segments and loops as they are parsed
- List of segments and loops as they are generated

NOTE: When you also check the **Trace Omitted Definitions**, you will see a chronological list of all segments as it checks against the input definition during parsing, indicated by the line in the report that begins with **...Looking** followed by the name of the segments. This additional option provides a complete and exact view of the process. The **Trace Segments** option only shows the segments that actually exist in the input, and typically produces a shorter report. For more information about the content of the segments, check the option, **Trace Elements**.

Trace Elements (Report Options)

Trace Elements prints the information about the process MW Translator follows when parsing and generating composite elements and elements for wrappers and documents. It shows those composites or elements as they appear in the input and output including the following:

- List and contents of elements and composites as they are parsed
- List and contents of elements and composites as they are generated

This shows the actual data after it is parsed but before it is stored in memory, in the data element store (DES). This also shows the data after it is generated for output. To view the data as it is stored internally after parsing and before generation, select the **Print DES** option.

Trace Omitted Definitions (Report Options)

Trace Omitted Definitions prints the information about the process MW Translator follows when parsing wrappers and documents. When used with the **Trace Segments** option, it shows all segments in the definition, as MW Translator attempts to match the input data with the definition. When used with the **Trace Elements** option, it shows all elements in the definition. Without this option, **Trace Segments** or **Trace Elements** only shows those segments and elements that actually appear in the input data.

HyperText Color

The **HyperText Color** option allows you to select a color to display the text on a maintenance window on which you can click to go to another window. This is very useful, because you immediately know when you can hotkey to another location. Be sure to select a color that contrasts well with the window background. If you select silver for a gray window background the text becomes invisible.

Reported Exceptions Limit

This field allows you to limit the number of errors that are reported on a translation report for the workbench. The maximum number you can enter is 9999. This selection does not affect the exceptions reported to backward acknowledgments. All aborts are reported without regard to this limit.

Procedures (Modify Options, General Tab)

The following procedures show how to control information that is printed on the processing reports.

To Generate Processing Reports

- 1 From the **File** menu, select **Options**.
- 2 From the Modify Options window, select the **General** tab.
- 3 Check **Always print reports**.
- 4 Optionally check any of the following boxes:
 - a) **Print DES** prints the locations and the contents of the data element store, which contains the input document, wrapper, and status, statistics, and errors to be reported in backward control documents.
 - b) **Print File Names** prints the names of the files used or sought during processing.
- 5 For debugging information, check **Print Processing Trace** and then any of the following boxes:
 - a) **Trace Wrappers** prints information about the input and output wrappers. For more information, refer to the topic, *Trace Wrappers (Report Options)* (on page 510)
 - b) **Trace Documents** prints information about the input and output documents. For more information, refer to the topic, *Trace Documents (Report Options)* (on page 510).
 - c) **Trace Segments** prints information about the input and output segments and loops. For more information, refer to the topic, *Trace Segments (Report Options)* (on page 511).
 - d) **Trace Elements** prints information about the input and output elements and composites. It also shows the contents of the elements. For more information, refer to the topic, *Trace Elements (Report Options)* (on page 511).
 - e) **Trace Omitted Definitions** must be used with **Trace Segments** or **Trace Elements**. It prints information about all input segments or elements in the definition, whether input data exists. For more information, refer to the topic, *Trace Omitted Definitions (Report Options)* (on page 511).

To Suppress Processing Reports

To suppress processing reports, which will only generate when there are errors or print statements in a map, proceed as follows:

- 1 From the **File** menu, select **Options**.
- 2 From the Modify Options window, select the **General** tab.
- 3 Check the option, **Only print reports if errors**.

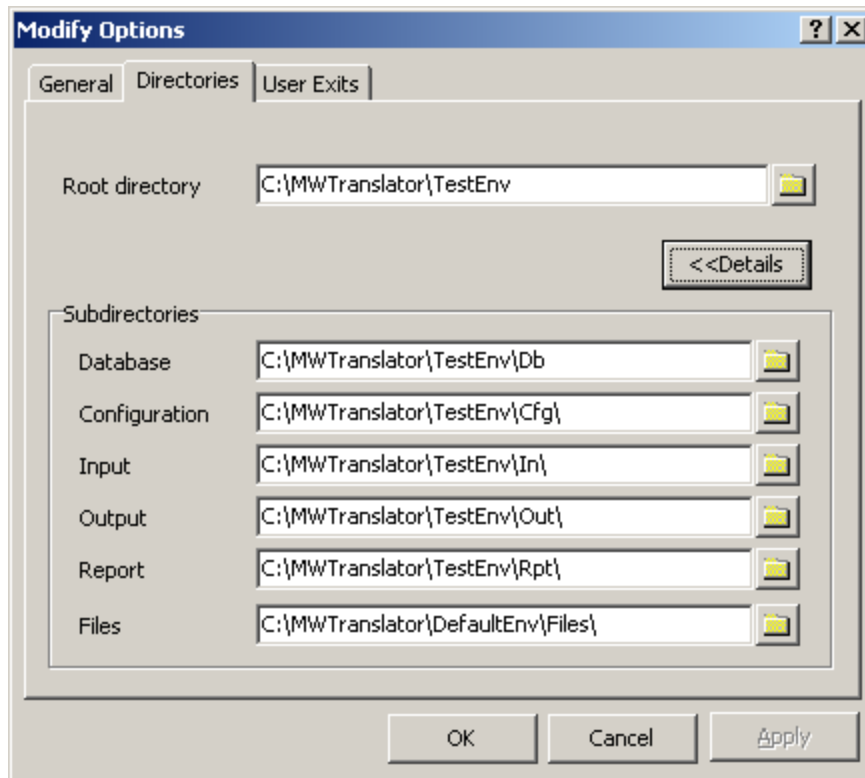
To Limit the Number of Exceptions Reported on the Translation Report

This will limit the number of errors reported on a translation report, in case you generate a large number of errors processing the input file. This limit does not affect the errors reported to a backward acknowledgment.

- 1 From the **File** menu, select **Options**.
- 2 From the Modify Options window, select the **General** tab.
- 3 Enter a value with a maximum of four digits that represents the greatest number of error messages you want reported on your translation report.

(Modify Options) Directories Page

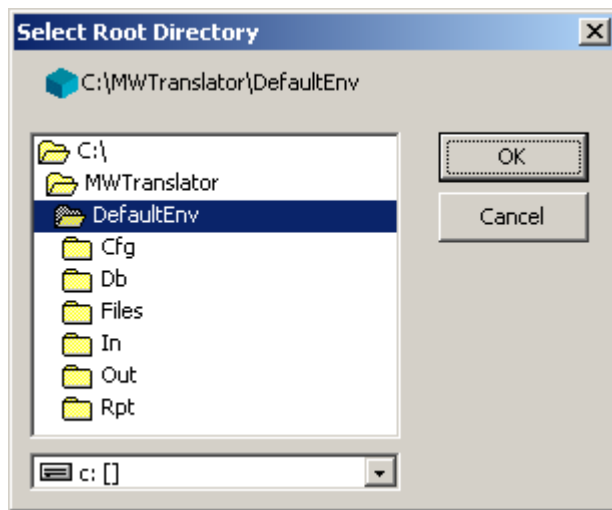
The **Directories** tab of the **Modify Options** dialog box allows you to specify the pathname for the database files, the configuration files, the input files, the output files, the reports, and the transfer files for the current Workbench environment. The Workbench generates the file names automatically.



Directories Page (Modify Options Dialog Box)

Root Directory

This identifies the location of the database files for this environment. This information is used for the Workbench configuration tasks. It extracts information from these files to create text files in the configuration subdirectory, which are then used for testing and production. When you change this value, you change the path for all subdirectories. When you click the browse button, the **Select Root Directory** dialog box appears.



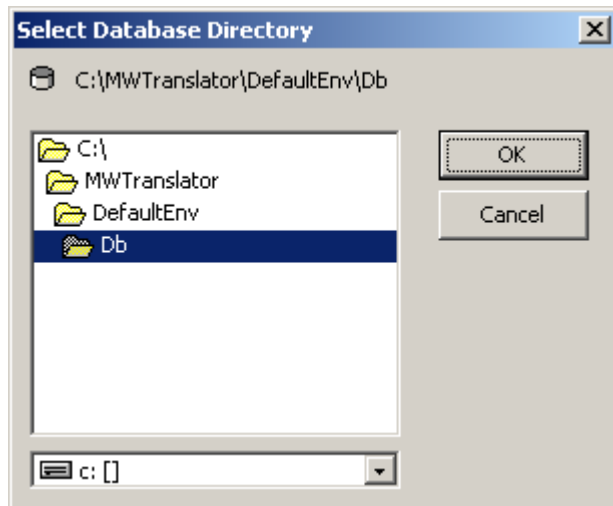
Select Root Directory Dialog Box

Details Button

This button allows you to display and then change each of the default subdirectory paths for this environment.

Database (Subdirectories)

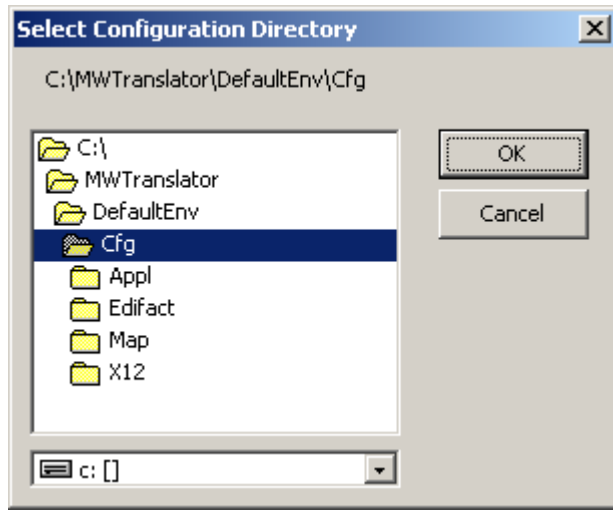
Enter or select the default location of the database files for this environment. When you select the browse button, the **Select Database Directory** dialog box appears.



Select Database Directory Dialog Box

Configuration (Subdirectories)

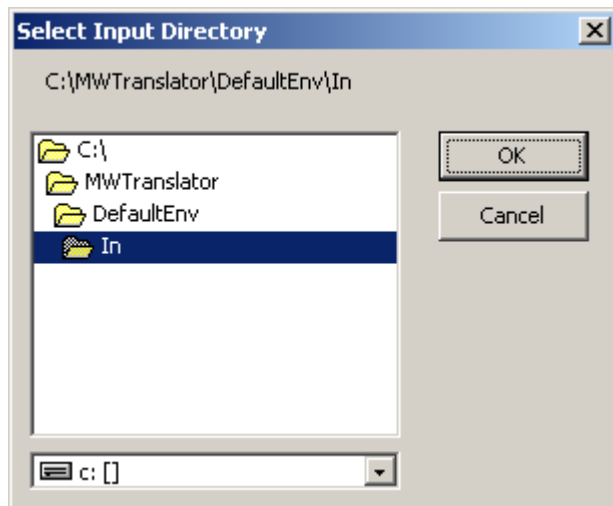
This identifies the location of all of the generated text files for this environment you will need for processing, such as, standards identification file, partner file, profile files, and subdirectories for maps, application standard definitions, X12 standard definitions, and EDIFACT standard definitions. When you select the browse button, the **Select Configuration Directory** dialog box appears.



Select Configuration Directory Dialog Box

Input (Subdirectories)

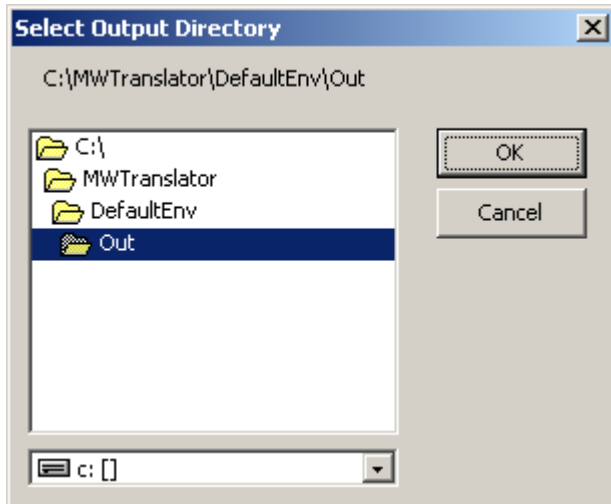
Enter or select the default location of the input data for this environment. When you select the browse button, the **Select Input Directory** dialog box appears.



Select Input Directory Dialog Box

Output (Subdirectories)

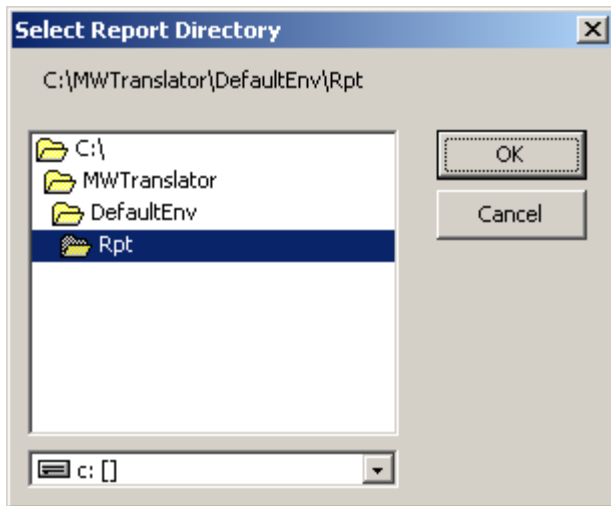
Enter or select the default location of the output data for this environment. When you select the browse button, the **Select Output Directory** dialog box appears.



Select Output Directory Dialog Box

Report (Subdirectories)

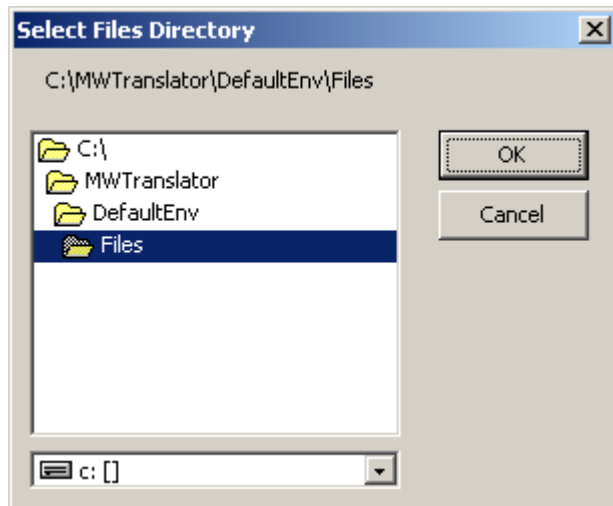
Enter or select the default location of the reports for this environment. When you select the browse button, the **Select Report Directory** dialog box appears.



Select Report Directory Dialog Box

Files (Subdirectories)

Enter or select the location of the exported transfer files for this environment. When you select the browse button, the **Select Files Directory** dialog box appears.



Select Files Directory Dialog Box

Procedures (Modify Options, Directories Tab)

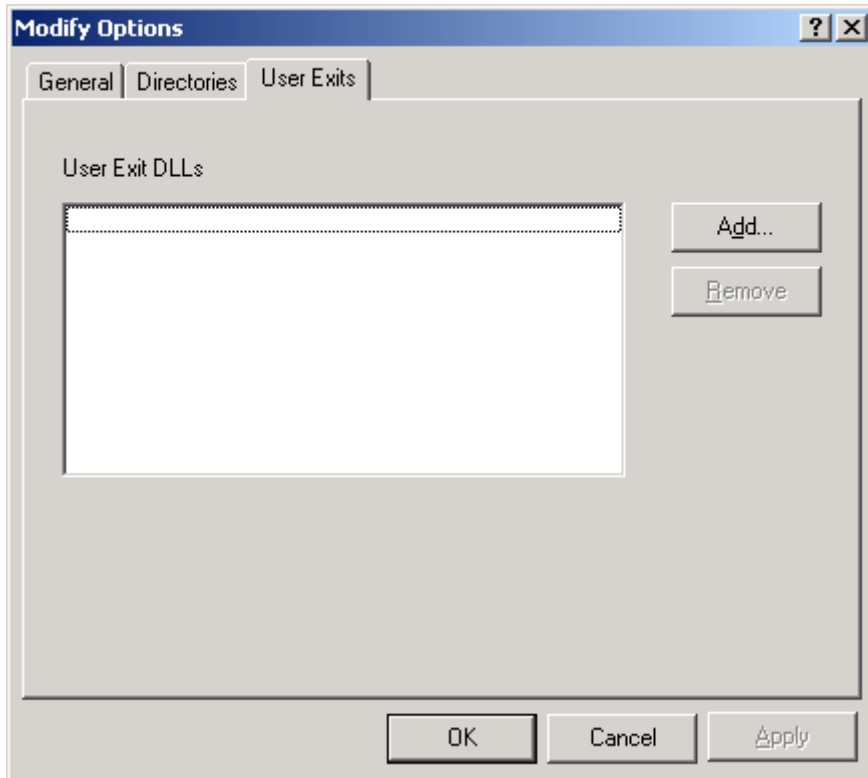
The following procedure changes the directory of the database, configuration, input, output, report or transfer files for the current environment.

To Change Database, Configuration, Input, Output, Report or Transfer File Locations for the Workbench

- 1 From the **File** menu, select **Options**.
- 2 From the **Modify Options** dialog box, select the **Directories** tab.
 - To change the directory for all subdirectories, proceed to step 5.
 - or –
 - To change only one or some of the directory locations, proceed to step 3.
- 3 Select the **Details** button.
- 4 Select the location from the appropriate box, depending on whether you are changing the location to which the Workbench is pointing for the database, configuration, input, output, report or files.
- 5 To change all pathname locations at once, select the location from the **Workbench root directory** box.
- 6 Close the window.

(Modify Options) User Exits Page

The **User Exits** tab of the **Modify Options** dialog box allows you to add and remove user exits that you will be using during processing. For more information about user exits, refer to the *MW Translator User Exits Programming Manual*.



User Exits Page (Modify Options Dialog Box)

User Exit DLLs

This box displays any user exit DLL files that may be invoked for this environment. You add or delete these file names using the **Add** and **Remove** buttons. You may define up to 9 user exit DLLs. Each DLL may have an unlimited number of user exits. There is an additional limitation specific to audit user exits: you may use only 8, and all audit user exits that are defined are called for each audit event. Other user exits have no such limitation.

Procedures

All user exits required for any processing in the current environment must be listed on the **User Exits** tab of the **Modify Options** dialog box. The following procedures maintain this list. Refer to the **User Exit DLLs** field definition for limits.

To Add a User Exit to the List for an Environment

- 1 From the **File** menu, select **Options**.
- 2 From the **Modify Options** dialog box, select the **User Exits** tab.
- 3 Select the **Add** button.
The **Select User Exit DLL** dialog box appears.
- 4 Select the user exit file that contains the user exit you need.
- 5 Select the **Open** button.
The user exit appears on the list.
- 6 Close the window.

To Delete a User Exit from the List for an Environment

- 1 From the **File** menu, select **Options**.
- 2 From the **Modify Options** dialog box, select the **User Exits** tab.
- 3 Select the DLL on the list you wish to delete.
- 4 Select the **Remove** button.
The user exit disappears from the list.
- 5 Close the window.

Pack Database Command

You access database maintenance from the **Tools** menu.

Pack Database

When you pack your databases the Workbench recovers space taken by deleted records. Selecting this command packs the database of the current environment, which is listed on the toolbar.

Procedures (Pack Database Command)

You can pack your database to free space to the operating system.


To Pack Your Database

The Workbench packs your current database. Make sure all windows except Data Explorer and Partner Explorer are closed. To point to another database, you must do so using the **Directories** page from the **Modify Options** dialog box.

- From the **Tools** menu, select **Pack database**.
The Workbench displays an information box and completes the process.

Print Command

A generic **Print** dialog box appears when you select an appropriate entity, such as a document or map, and

then select the **Print** button  from the toolbar or **Print** from the **File** menu.

The following table provides a list of possible reports, and the entity to select in order to obtain the report.

Report Type	Entity to Select	Content of Report
Document	One or more documents	All selected documents, with option to include elements, segment conditions, and Edibasic element validation code
Wrapper	One or more wrappers	All selected wrappers, with option to include elements, segment conditions, and Edibasic element validation code
Maps	One or more maps	All selected maps
Trade Agreement Profiles	Trade Agreements folder One or more trade agreement profiles	All trade agreement profiles All selected trade agreement profiles
Acknowledgment Profiles	Acknowledgments folder One or more acknowledgment profiles	All acknowledgment profiles All selected acknowledgment profiles
Cross-References	Cross-References folder One or more cross-reference tables	All cross-reference profiles

Report Type	Entity to Select	Content of Report
Locations/StdID	Location in Locations/StdID folder One or more wrappers for a location	All Standard ID wrappers defined for the location All wrappers selected for the location
Partner	Partners folder One or more partners	All partner profiles All selected partner profiles
Partner Relationship	Partner Relationships folder One or more partner relationship	All partner relationships All selected partner relationship profiles

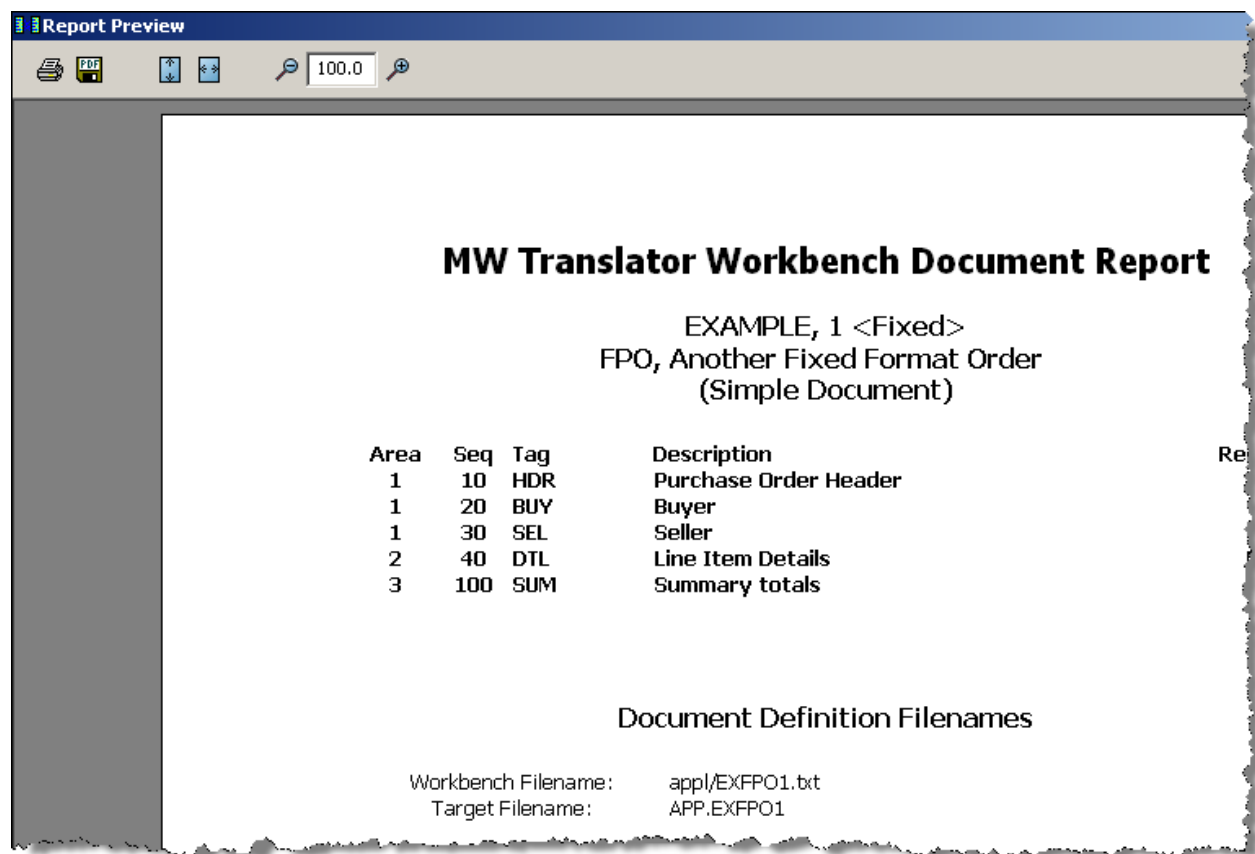
Print Preview Command

The Report Preview window appears when you select an appropriate entity and then select the **Print**




Preview button from the toolbar or **Print Preview** from the **File** menu. From here, you can print to a printer or to a PDF file, change the zoom and page through the report.

NOTE: To print part of a report, you should print to a PDF file and print selected pages from the PDF viewer.




Print to PDF Command

When you select an appropriate entity, such as a document or map, and then select the **Print to PDF** button  from the toolbar or **Print to PDF** from the **File** menu, a generic **Save As** dialog box appears.

For a list of reports, refer to the topic, *Print Command* (on page 522).

Select Environment Command

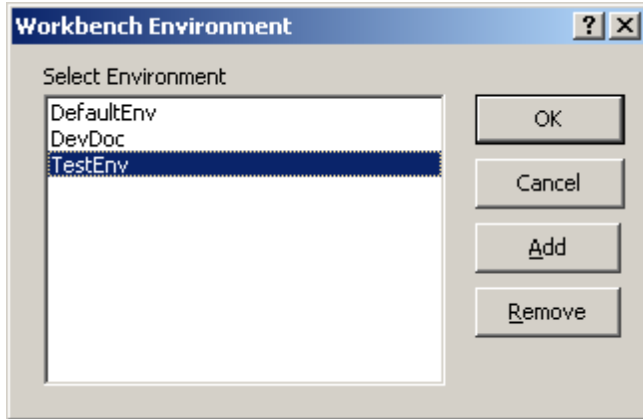
When you first install MW Translator, you will install a Workbench, an Operator Program and a database environment. Use the install program to install additional database environments. This allows you to install one version of the Workbench software and have multiple database environments to use for testing.

The **Select Environment** button  allows you to change the environment to which the Workbench points.

An environment directory contains the following subdirectories and files:

Subdirectory	Description of Files
CFG	Text files used by the TRM that contain multiple definitions for partner relationships, partners, acknowledgment and trade agreement profiles, locations and standard identification, and cross-references.
APPL	Text files for each proprietary document and wrapper
EDIFAC	Text files for each EDIFACT document and wrapper
T	Text files for each map
MAP	Text files for each X12 document and wrapper
X12	
DB	Database configuration files
FILES	Transfer files used for import and export
IN	Input files for processing
OUT	Output files from processing
RPT	Processing reports

To add environments to or remove environments from the list, from **File** the menu select, **Select Environment** bar. This displays the **Workbench Environment** dialog box. From the list, select any existing environment. Add or remove database environments from the list using the **Add** and **Remove** buttons.



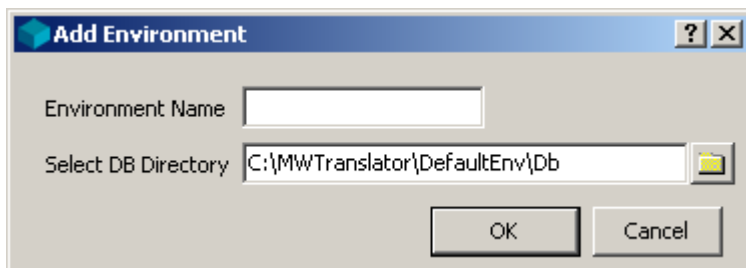
Workbench Environment Dialog Box

Select Environment

From the list, select any existing environment. Add or remove database environments from the list using the **Add** and **Remove** buttons.

Add Button

The **Add** button allows you to add an existing database to your list by assigning it an environment name. The environment includes all supporting directories, such as Cfg, In, Out, Rpt. When you select the **Add** button, the **Add Environment** dialog box appears.



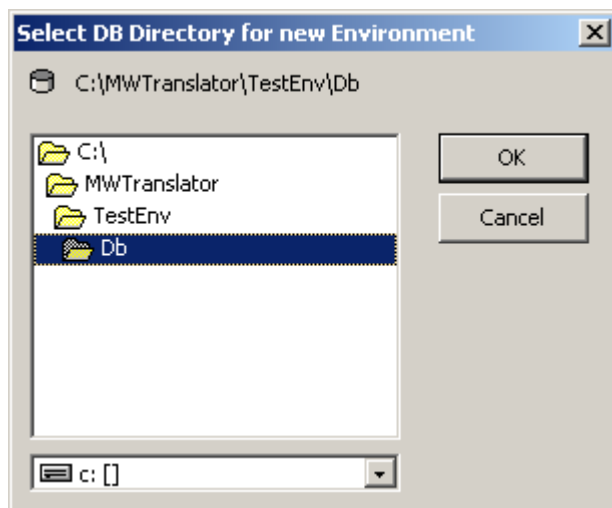
Add Environment

Environment Name

Enter a name for the environment. This name appears on the list of database environments for future selection.

Select DB Directory

Enter or select a path using the **Browse** button for the **DB** subdirectory. When you select the **Browse** button, the **Select DB Directory for new Environment** dialog box appears from which you can choose the appropriate directory.



Remove Button

The **Remove** button allows you to remove an environment from your list. It does not delete the directory or the files. You may not remove the last environment on the list.




Procedures (Select Environment Command)

The following procedures describe how to take advantage of one of the most powerful features of MW Translator, switching between database environments. This feature allows users to maintain separate test environments and run tests by pointing to the appropriate environment.

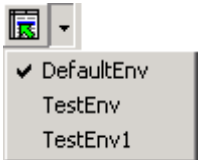
TIP: To install additional environments, use the Workbench install program and select **New Database Environment**.

To Add a Database Environment to the List

Any environments you want to add to the list must exist. This procedure does not create environments. When you select database environments created prior to version 5.0, you will be asked to first convert the database.



- 1 From the toolbar, select the leftmost part  of the **Select Environment** button . The **Workbench Environment** dialog appears.
- 2 Select the **Add** button. The **Add Environment** dialog box appears.
- 3 Use the **Browse** button  to locate the DB subdirectory of the environment you want to add. The location appears in the **Select DB Directory** box.
- 4 Enter a name for the environment for easy recognition.
- 5 Select **OK** to close the window. The environment name appears on the **Select Environment** list of the **Workbench Environment** dialog box.
- 6 Click **OK**.

The environment is now accessible from the Workbench toolbar **Select Environment** list





To Remove a Database Environment from the List

You may not remove the last environment from the list.

- 1 From the toolbar, select the leftmost part  of the **Select Environment** button . The **Workbench Environment** dialog appears.
- 2 From the **Select Environment** list, select the environment you want to remove.
- 3 Select the **Remove** button. The environment disappears from the list.
- 4 Select **OK**.

To Switch between Database Environments

- 1 From the toolbar, select the right most part  of the **Select Environment** button . A check mark appears next to the current environment.
- 2 From the list of environments, select the environment to which you want to point. The Workbench reconfigures itself to point to the new environment. The name of the current environment appears on the title bar.

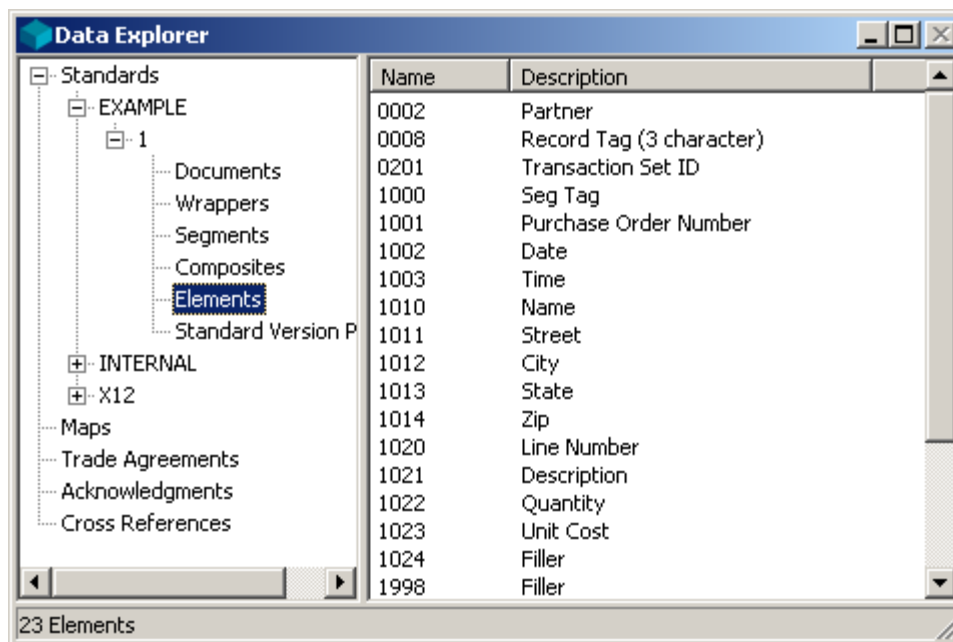
This page intentionally blank

Data Explorer Reference

This chapter contains reference information for all windows accessible from the Data Explorer window, including tasks that might be performed from the various windows.

Data Explorer Window

You access all but the standard identification and partnership definitions using the Data Explorer. To access partnership definitions you will use Partner Explorer. This window behaves like a typical browser window. The left pane displays all currently defined standards, maps, trade agreements, acknowledgments, and cross-references. The right pane displays the entities defined for a selection in the left pane.



Data Explorer Window

To Open a Properties Window

Properties windows provide additional configurations for definitions contained in the following folders:

- Documents
- Wrappers
- Segments
- Composites
- Elements
- Standard Version Profile
- Maps

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the appropriate definitions in the right pane.
- 2 In the right pane, double-click the definition whose properties you want to view.

The configuration window opens.



- 3 From the toolbar, select the **Properties** button

– or –

From the menu bar, select **File** and then **Properties**.

– or –

On the configuration window, right-click and select **Properties**.

To Open the Data Explorer Window

- From the menu bar, choose **View**, and then choose **Data Explorer**.

– or –

- From the toolbar, choose the **Data Explorer** button



Acknowledgment Profile Window

You must configure an acknowledgment profile when you want to return backward acknowledgments to the sender of the original message. You access this window by selecting an existing definition in the Data Explorer window or by adding a definition.

Acknowledgment Profile Name

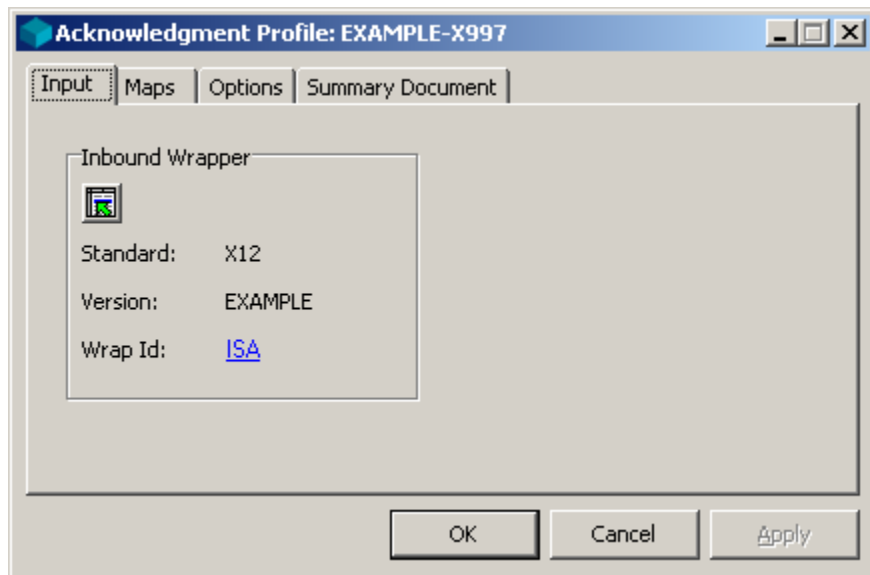
The name uniquely identifies the Acknowledgment profile. It may be from 1 to 32 alphanumeric characters. You enter the name in the **New Acknowledgment Profile** dialog box when you add a new profile.



New Acknowledgment Profile Dialog Box


(Acknowledgment Profile) Input

The **Input** tab allows you to choose the wrapper definition that the TRM will use to match against the incoming wrapper data. When there is a match, it will use this acknowledgment profile definition to generate an acknowledgment, as further specified on the **Maps**, **Options**, and **Summary Document** tabs.



Input Tab (Acknowledgment Profile Window)

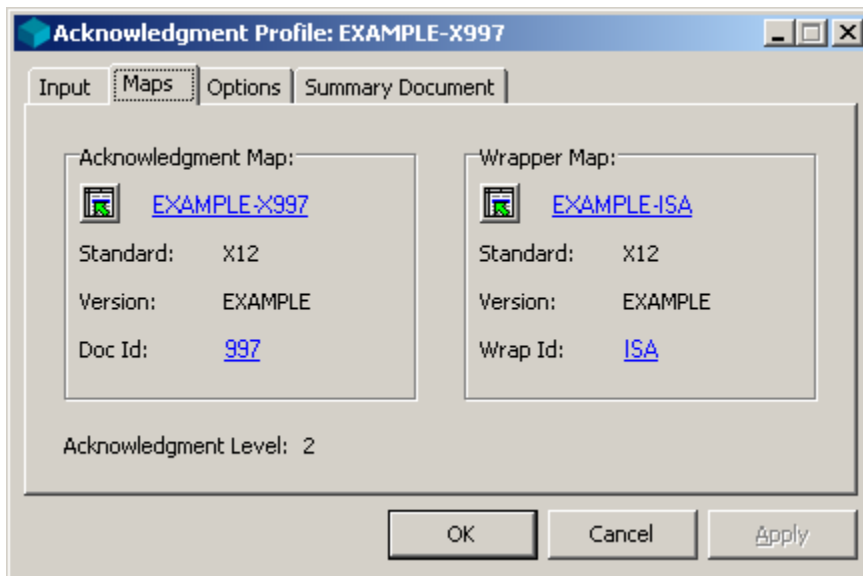
Inbound Wrapper

To receive acknowledgments, you must specify a wrapper definition. The wrapper of the incoming document identified during the standard identification process must match the standard and version given here. You can use the **Select** button  to choose a valid wrapper definition. You can also jump to the Wrapper window by clicking on the **Wrap Id** value.

NOTE: Fields that allow jumps are underlined and displayed in another color. You can change the hypertext color by selecting **File|Options|General tab** from the menu bar.


(Acknowledgment Profile) Maps

The **Maps** tab allows you to choose the maps that the TRM will use to generate the acknowledgment and its wrapper. This tab also displays the level at which the acknowledgment reports as defined on the Document Properties window.




Maps Tab (Acknowledgment Profile Window)

Acknowledgment Map

This option specifies the map that will be used to create the acknowledgment. You can use the **Select** button  to choose a map. To jump directly to the Map window, double-click a map name. To jump to the Document window click the **Doc Id** value.

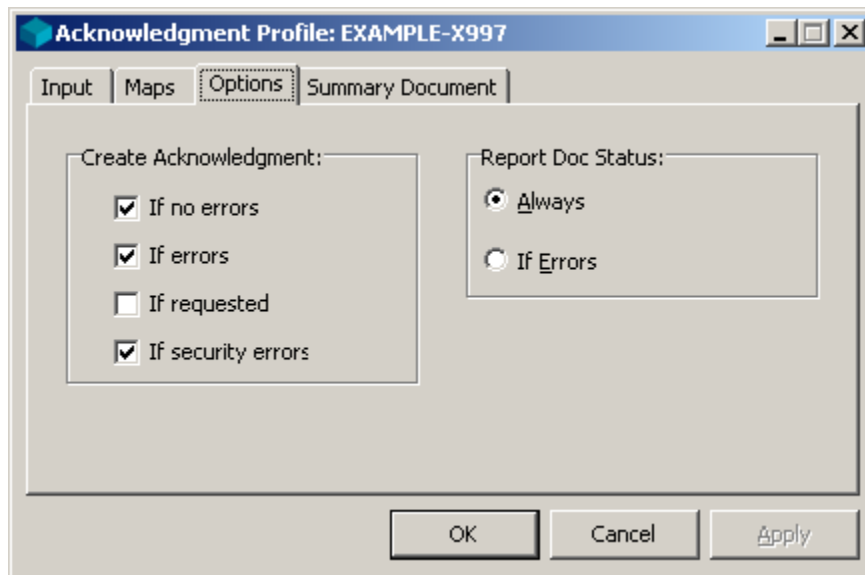
Wrapper Map

This option specifies the map that will be used to create the wrapper for the acknowledgment. You can use the **Select** button  to select a map. To jump directly to the Map window, double-click a map name. To jump to the Wrapper window, click the **Wrap Id** value.

(Acknowledgment Profile) Options

The **Options** tab allows you to specify when to create an acknowledgment and when to report the status of the document.

You will always generate an acknowledgment when you select **If no errors**, **If errors**, and **If security errors**.



Options Tab (Acknowledgment Profile Window)

Create Acknowledgment

Different standards have different requirements for creating acknowledgments. The choices here allow you to comply with these various requirements when you select one or more options that you need.

Condition	Description
If no errors	Generate an acknowledgment when there are no errors in the incoming documents.

Condition	Description
If errors	Generate an acknowledgment when there are errors in the incoming documents, not including security errors.
If requested	Generate an acknowledgment when the incoming wrapper contains any non-zero value for the request element. This element must be associated with the Acknowledgment Request internal field (identified in the Field column on the Segment window for that wrapper element).
If security errors	Generate an acknowledgment when there are errors returned from security processing, such as that required by EDIFACT AUTACK. Any condition that would not allow the user exit to return an accept response to the TRM will generate a security error. In addition, any condition where the expected summary document is not found will generate a security error.

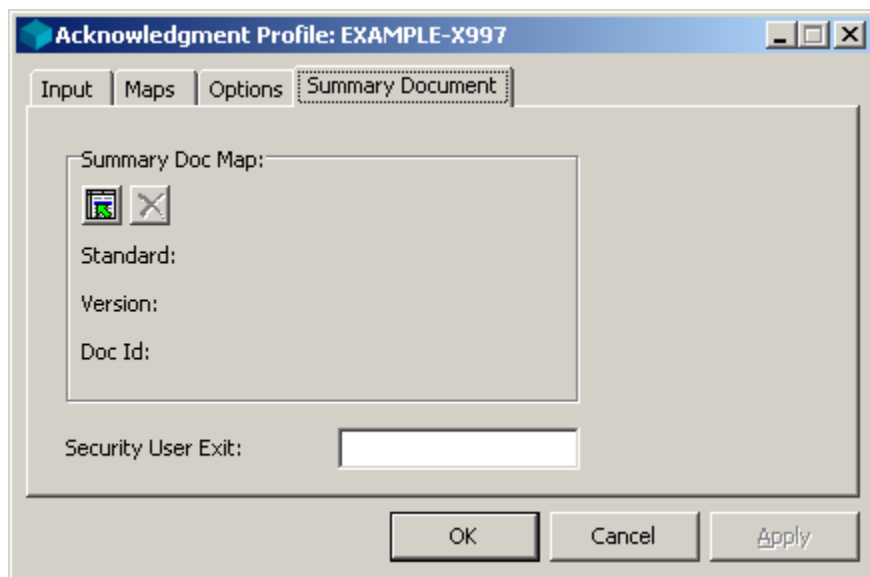
NOTE: If the TRM cannot find the identified security user exit, translation aborts with error **5001, Invalid user exit name**. There will be no output.

Report Doc Status

Different standards have different requirements for reporting the status of documents in acknowledgments. The choices here allow you to comply with these various requirements. When you choose **Always**, the TRM will always report the status of the documents, whether there were errors or not. When you choose **If Errors**, the TRM only reports the status of documents when they contain errors.


(Acknowledgment Profile) Summary Document


The **Summary Document** tab allows you to specify whether a summary document is to accompany the acknowledgment and to select the map for the summary document. A summary document responds to information in an interchange and is included in that interchange. You can also specify a security user exit that must be executed.



Summary Document Tab (Acknowledgment Profile Window)

Summary Document Map

When the TRM generates an acknowledgment and if a partner requires a summary document, such as AUTACK, to accompany the acknowledgment, such as CONTRL, then this map will be used to create the summary document. The TRM generates the summary document as the last document of the interchange. You select the appropriate map using the **Select Source** button  to browse your directory.

If you no longer want to use a summary document, deselect the map using the **Deselect** button .

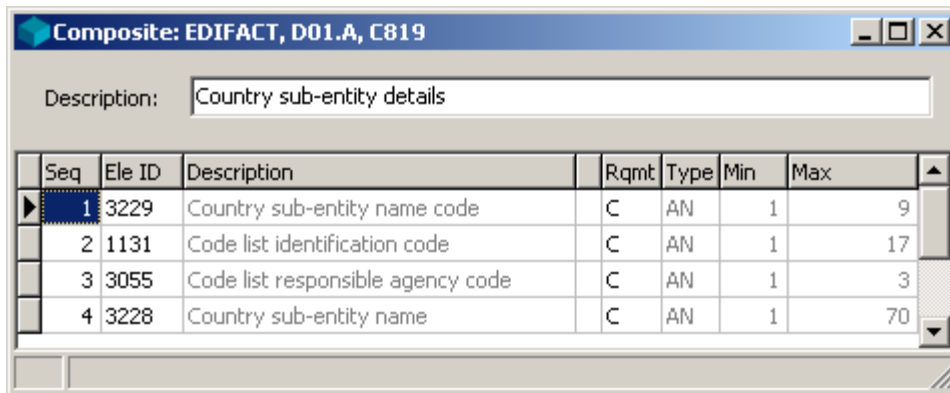
Security User Exit

If you need to implement a security procedure that executes before the TRM creates the acknowledgment, such as that required for EDIFACT AUTACK, you must do so with a user exit. You identify that user exit here. It must match the registered name you specify as the **name** parameter of the **ERMRegisterSecExit** function of your user exit.

The security user exit is typically used to calculate hash totals for secured documents. For more information about defining user exits in general, and security user exits in particular, refer to the *MW Translator User Exits Programming Manual*.

Composite Window

The Composite window allows you to enter information about composite elements. A composite element is a logical device used to relate element information as a group of component elements. The Composite window is used to identify the composite ID, and the components that are part of the composite. You can also specify a different parsing and generation technique at the composite element level. In addition, you can specify user validation routines to augment the basic compliance validation during wrapper and document parsing.



Seq	Ele ID	Description	Rqmt	Type	Min	Max
1	3229	Country sub-entity name code	C	AN	1	9
2	1131	Code list identification code	C	AN	1	17
3	3055	Code list responsible agency code	C	AN	1	3
4	3228	Country sub-entity name	C	AN	1	70

Composite Window

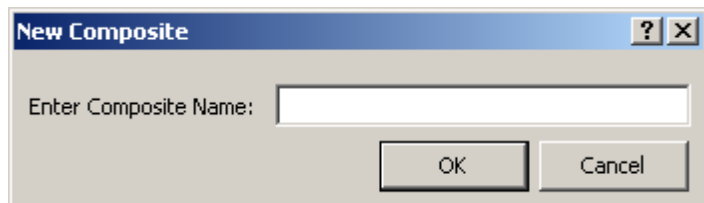
To access this window:

- In the right pane of the Data Explorer, select an existing definition
 - or –

- In the left pane, select **Composites** within your standard and version, right-click your mouse, and then select **Add**

The Composite Element has an associated *Composite Properties window* (on page 549). To access the Composite Properties window, refer to the topic, *To Open a Properties Window* (on page 532).

When you add a new definition, the **New Composite** dialog box appears.



New Composite Dialog Box

Composite Name

The composite name together with the standard and version uniquely identify a composite element. The name may contain from 1 to 8 alphanumeric characters.


Description

The description can contain from 1 to 80 alphanumeric characters. It is an optional value for you to use to describe the composite element.

Seq (Components)

The sequence field contains the numeric value that determines the sequence of the component element within the composite.

Ele ID (Components)

The element ID represents the reference number (X12) or the element tag (EDIFACT) for that element of the composite. The element ID may be from 1 to 8 alphanumeric characters. To select an existing element, click twice slowly, and when the selection button, , appears, click it to display a list of existing elements from which you can choose.

Description (Components)

The description is display-only, indicated by dimmed field values. You access this value in the Element window.

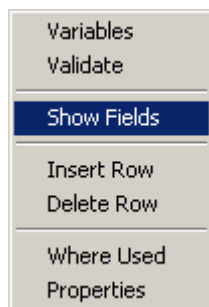
Blank Column

This narrow column with the blank header contains a check mark whenever you have attached Edibasic validation routines to this entity. This column allows you to see at a glance, which elements have validation routines.

Field (Components)

This column displays any internal fields that have been assigned to a particular component element. It is used most often for elements in wrapper segments, but may also be used for elements in document segments. When the TRM parses the incoming wrappers, the element values associated with these internal fields are used to find partner profiles and do compliance checking.

The column is visible for composite elements used in document segments only when a component element has been associated with an internal field. To display the column, right-click and select **Show Fields** from the menu.



For instructions to associate an internal field with a component element, refer to the topic, *To Assign an Internal Field To a Component Element* (on page 545).

The following example shows internal fields that are assigned to some elements of the EDIFACT UNB wrapper segment.

Segment: EDIFACT, 2, UNB
Description: Interchange header

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	S001	SYNTAX IDENTIFIER		M			
2	S002	INTERCHANGE SENDER		M			
3	S003	INTERCHANGE RECIPIENT		M			
4	S004	DATE/TIME OF PREPARATION		M			
5	0020	Interchange control reference	Control Reference	M	AN	1	14
6	S005	RECIPIENTS REFERENCE, PASSWORD		C			
7	0026	Application reference		C	AN	1	14
8	0029	Processing priority code		C	A	1	1
9	0031	Acknowledgement request	Acknowledgement F	C	N	1	1
10	0032	Communications agreement id		C	AN	1	35
11	0035	Test indicator	Test Indicator	C	N	1	1

The next example shows internal fields that are assigned to the EDIFACT S002 Composite element for the interchange sender.

Composite: EDIFACT, 2, S002
Description: INTERCHANGE SENDER

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	0004	Sender identification	Send Partner ID	M	AN	1	35
2	0007	Partner identification code qualifier	Send Partner Qual	C	AN	1	4
3	0008	Address for reverse routing	Send Partner IntID	C	AN	1	14

Rqmt (Components)

The requirement field can be one of the following values, which you may choose from a menu if you click slowly twice in the cell. Remember that if you do a fast double-click, you jump to the Element window:

- M** Mandatory. This component element must appear in the composite element. If it does not, the TRM generates a compliance error.

- O** Optional. This component element is not required in the composite element.
- C** Conditional. (This is equivalent to X for Relational in X12.) The required presence of this component element depends on the value or presence of one or more elements within the composite.

NOTE: EDIFACT makes no distinction between conditional and optional. It uses the designation of conditional for optional elements, with or without conditions.

Type (Components)

This contains the data type, used for formatting the data. This field is display-only. You access this value in the Element window.

Min (Components)

This contains the minimum element length. This field is display-only. You can access this value in the Element window.

Max (Components)

This contains the maximum element length. This field is display-only. You can access this value in the Element window.

Procedures (Composite)

The following procedures describe how to define composite elements.

To Add a Composite Element

Make sure the Data Explorer window is *open* (on page 532).

- 1** In the left pane, expand the folders to view the appropriate standard and version, and then select **Composites**.
- 2** From the **File** menu, select **Add**.
The **New Composite** dialog box appears.
- 3** Enter the *name* (on page 539) of your new composite element, using 1 to 8 alphanumeric characters, and select **OK**.
A blank Composite window appears
- 4** Enter a description for your composite, and add component elements to the composite. For instructions, refer to the topic, *To Add a Component Element to a Composite* (on page 544).
- 5** After you have added component elements to your composite, do one of the following:

- Close the window to save the information

– or –

- To add another element:

- Press **CRTL+A**

– or –

- Select **Add** from the **File** menu.

The **New Element** dialog box re-appears and you can repeat steps 3-5.

To Modify an Existing Composite Element

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the appropriate standard and version, and then choose **Composites**.
- 2 In the right pane, from among the list of composite elements, double-click the name of the composite element you want to modify.
- 3 Make changes, and close the window.

To Save a Composite Element


The Workbench automatically saves records when you exit the window. You can also make an interim save by choosing **Save** from the **File** menu.

To Delete a Composite Element

Before you delete definitions, always check the **Where Used** list to determine how this deletion will affect other entities.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the appropriate standard and version, and then select **Composites**.
- 2 In the right pane, select the one you want to delete.
- 3 To view where the composite is used, right-click and select **Where Used** from the menu.

- 4 From the toolbar, select the **Delete** button  .

A **Confirm** dialog box appears.

- 5 Select **OK**.

To Sequentially Display Definitions of Composite Elements

This procedure displays all definitions from all standards. The standard and version will change when you display definitions from a different standard. From the Composite window you can page backward and forward within the definitions:

- 1 From the **View** menu, select **Next** or press **F8** to review the next composite definition.
- 2 From the **View** menu, select **Prior** or press **F7** to review the previous composite definition.

To Add a Component Element to a Composite

Make sure the Data Explorer window is *open* (on page 532).


- 1 If you are already in the Composite window, proceed to step 2.
– otherwise –
From Data Explorer, choose the composite to which you want to add components.
- 2 Place your cursor in the components detail area, which is the box below the header area.
- 3 Create a blank line.

IMPORTANT: You may insert a line anywhere in the composite. They are listed in order based on the unique sequence numbers you use. To insert components, you must have a gap in your sequence numbers. If you have no gap in your sequence numbers, you must manually resequence other components to create a gap.

- To add a line at the end, place your cursor in the last line and press the **Tab** key or the down-arrow key.
– or –
 - To insert a line before another line, place your cursor on the detail line, and
 - Press the **Insert** key
– or –
 - Right-click, and select **Insert Row** from the menu
- 4 In the blank line, enter the appropriate information, which may include creating a new element.

NOTE: When you move your cursor to a different line, the Workbench will display the components in the correct order.

- a) In the **Seq (on page 539)** column, type a sequence number.
- b) To add a new element:
 1. In the **Ele ID** (on page 539) column, type the element ID, press **Tab**, and then double-click quickly.
 2. When the Element window appears, *add an element* (on page 590).
- c) To select an existing element:

1. In the **Ele ID** (on page 539) column, right-click once.
 2. Click the selection button, , when it appears.
 3. From the **Select Element** dialog box, select the appropriate element and click **OK**.
- d) In the **Rqmt** (on page 541) column, type or select the requirement designator.
5. Close the window to save your changes.

To Modify an Existing Component Element

Make sure the Data Explorer window is *open* (on page 532).

1. Place your cursor in the components detail area using the **Tab** key or right-click.
2. If you are not already positioned at the desired field, use the **Tab** key or the arrow keys to position the cursor to the field where you want to make the change. Type in the appropriate information.

To Delete a Component Element From a Composite Element

When you delete the component element from the composite definition, you do not delete the element itself, you simply remove it from the composite.

Make sure the Data Explorer window is *open* (on page 532).

1. In the left pane, expand the folders to view the appropriate standard and version, and then choose **Composites**.
2. Place your cursor in the components box using the **Tab** key.
– or –
Right-click the mouse in the components box.
3. Use the **Tab** key or the arrow keys to position the cursor to the line that you want to delete.
4. Right-click and select **Delete Row** from the menu.
A **Confirm** box appears.
5. Select **OK**.

To Assign an Internal Field To a Component Element

You can tell the TRM to store certain information for future reference during mapping when it parses the incoming data, usually from a wrapper. You do this by assigning a predefined internal field location to an element in a Composite window.

Make sure the Data Explorer window is *open* (on page 532).

1. In the left pane, select **Composites** from the expanded view of the appropriate standard and version.
2. From among the list of composite elements in the right pane, double-click the name of the composite to whose component element you want to assign an internal field.

The Composite window appears.

3 In the detail area of the Composite window, right-click to display the menu.

4 From the menu, select the **Show Fields** command.

The **Field** column appears.

5 In the **Field** column of the target component element, click twice slowly. (If you double-click quickly, you will go to the Element window.)

6 Click the down arrow to display the list of internal fields.

7 Scroll through the list and select the appropriate field for the element.

8 Close the window. The Workbench automatically saves the changes.

To Add Edibasic Validate Routines to Component Elements

You can add your own validate routines to component elements. Although the TRM does compliance checking, you may want to perform additional validations. The TRM will execute the routine during parsing only when this component occurs within this composite in this document. The validate routine may override another validate routine written for this element from the Element window. You should always review validate routines by printing a document report, so you will know which ones will take effect.

Make sure the Data Explorer window is *open* (on page 532).

1 In the left pane, expand the folders to view the appropriate standard and version, and select **Composites**.

2 In the right pane, double-click the composite element to whose component element you want to add the variable.

The Composite window appears.

3 Place your cursor in the component box, and select the component to which you will attach the validation routine.

4 Right-click and select **Validate**.

The Edit window appears with **Method Validate** header and trailer code.

5 Enter your validate routine between the header and trailer code, which typically includes using the **Exception** function.

6 *Check your syntax for errors* (on page 585).

7 Select **OK**.

The Composite Element window appears with a check mark in the column for the component indicating user code exists for it.

8 *Print a document report* (on page 571) to review the validate routines that may be invoked for the document.

To Declare Variables for Validate Routines on Composite Elements

You can declare local variables or global variables to use within validate routines for composite elements. You can declare local variables on the **Variables** tab or within the method on the **Validate** tab. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the **Variables** tab only. They have a lifetime from the point of declaration to the end of processing for the input stream. For example, you can store data based on validate routines during the parsing process and use the stored data later to create a proprietary acknowledgment.

Make sure the Data Explorer window is *open* (on page 532).

To create local or global variables for a composite element, proceed as follows:

- 1 In the left pane, expand the folders to view the appropriate standard and version, and then choose **Composites**.
- 2 In the right pane, double-click the composite element for which you want to create the variable.
The Composite window appears.
- 3 Place your cursor in the window header area (top part of the window).
- 4 Right-click and select **Variables**.
The Edibasic Edit window appears.
- 5 On the **Variables** tab, enter your local or global variable declaration statement.
The following example declares a global variable:

```
Global Gmyvar as Integer
```


The following example declares a local variable:

```
Dim Lmyvar as Integer
```
- 6 **Check your syntax for errors** (on page 585).
- 7 **Print a document report** (on page 586) to review the validate routines that may be invoked for the input document.

To Declare Variables for Validate Routines on Component Elements

You can declare local variables or global variables to use within validate routines for component elements within a specific composite element. You can declare local variables on the Variables page or within the method on the Validate page. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the Variables page only. They have a lifetime from the point of declaration to the end of processing of the input stream. For example, you can store information based on validation routines during the parsing process and use it later to create a proprietary acknowledgment.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Composites** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the composite element for which you want to create the variable.

- 3 Position your cursor in the component detail area and select the appropriate component element.
- 4 Right-click and select **Variables**.

The Edibasic Edit window appears with the Variables page selected.

- 5 Enter your local or global variable declaration statement here. The following example declares a global variable:

```
Global Gmyvar as Integer
```

The following example declares a local variable:

```
Dim Lmyvar as Integer
```

- 6 *Check your syntax for errors* (on page 585).
- 7 Select **OK**.

When you close the window, a check mark appears in the blank column for that component indicating user code exists for it.

- 8 *Print a document report* (on page 571) to review the validate routines that may be invoked for the document.

To Add Edibasic Validate Routines to Composite Elements

You can add your own validate routines to composite elements. Although the TRM does compliance checking, you may want to perform additional validations. The TRM executes the routine during parsing whenever this composite is used in a segment. The validate routine may be overridden by another validate routine written for this composite within a Segment window. You should always review validate routines by printing a document report, so you will know which ones will take effect.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Composites** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the desired composite element.

The Composite window appears.

- 3 Place your cursor in the Composite window header area (top part of the window).
- 4 Right-click and select **Validate**.


The Edibasic Edit window appears with Method Validate header and trailer code.

- 5 Enter your validate routine between the header and trailer code, which typically includes using the **Exception** function.
- 6 *Check your syntax for errors* (on page 585).
- 7 *Print a document report* (on page 571) to review the validate routines that may be invoked for the document.

Composite Properties Window

The Composite Properties window allows you to define some additional characteristics of a composite element, such as category, purpose and conditions. It also displays the list of segments where the composite is used, in case you need to make changes.

To access the Composite Properties window from the Composite window, select **Properties** from the **File**

menu or select the **Properties** button  from the toolbar.

Procedures (Composite Properties)


These procedures affect conditions for composite elements.

To Add a Condition to a Composite

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Composites** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the composite to which you want to add conditions.

The Composite window appears.

- 3 From the toolbar, click the **Properties** button, .
The Composite Properties window appears.
- 4 Select the **Conditions** tab, place your cursor in the detail area, and proceed as follows:
 - a) In the **Seq** column, enter a sequence number, which typically is the same as the sequence number of the component element within the composite to which the condition applies.
 - b) To enter multiple conditions for a component element, in the **Sub Seq** column, enter a number that together with the **Seq** number uniquely identifies the condition.
 - c) In the **Condition** column, click twice slowly to display an arrow, and click the arrow and choose one of the types of conditions from the list:

P	Paired Condition	If any data element specified is present, then all must be present.
R	Required Condition	At least one of the data elements specified must be present.

E	Exclusion Condition	Not more than one of the data elements specified may be present.
C	Conditional Condition	If the first data element specified is present, then all others must be present.
L	List Conditional Condition	If the first data element specified is present, then at least one of the others must be present.

- d) In the blank columns, enter the sequence numbers of the component elements to which the conditions apply. They must be the same as the sequence numbers in the elements detail area in the Composite window.
- e) To add more conditions, tab to create a new row and repeat steps a-e.

5 Select **OK** to close the window to save the information.

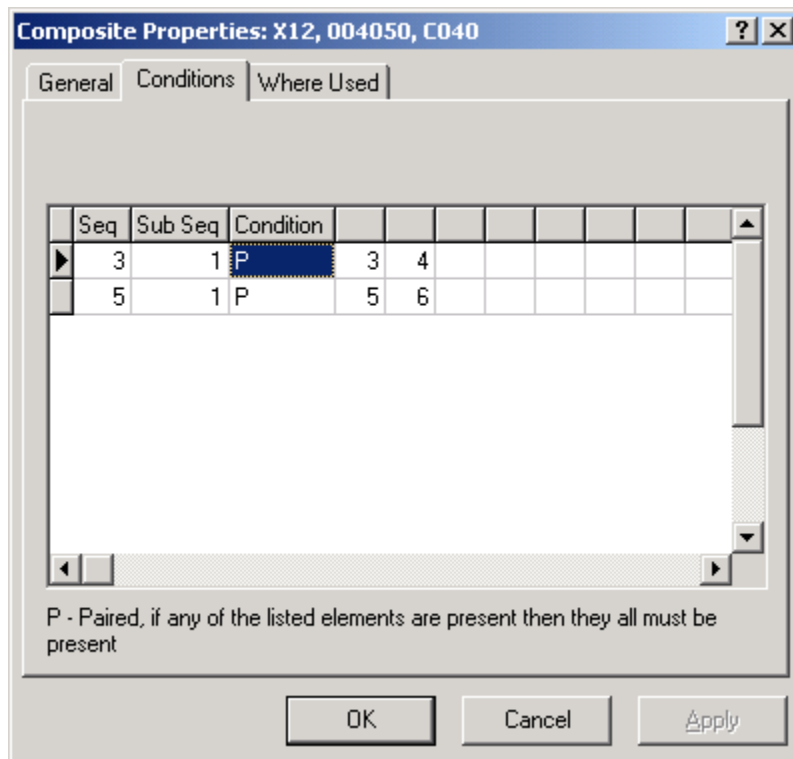
To Delete a Condition From a Composite

Make sure the Data Explorer window is *open* (on page 532).

- 1** In the left pane, select **Composites** from the expanded view of the appropriate standard and version.
- 2** In the right pane, double-click the name of the composite whose condition you want to delete.
- 3** From the **File** menu, select **Properties**.
The Composite Properties window appears.
- 4** From the Composite Properties window, select the **Conditions** tab.
- 5** Select the condition you want to delete.
- 6** Press **CTL+DELETE** from the keyboard.
A **Confirm** dialog box appears.
- 7** Select **OK**.
- 8** Select **Apply** or **OK** to save the change.

(Composite Properties) Conditions

The **Conditions** tab allows you to enter information about elements that are part of this composite. The TRM will compare the incoming data with these conditions for compliance. Note that when you select the condition, a description appears below the list box. To delete a condition, select the condition and press **CTRL+DELETE**.



Conditions Tab (Composite Properties Window)

Seq

This sequence number (Seq) together with the sub-sequence (Sub Seq) number uniquely identifies the condition. Typically, the sequence number is the same as the element within the composite to which the condition applies, although this is not mandatory.

Sub Seq

The sub-sequence (**Sub Seq**) number together with the sequence (**Seq**) number, uniquely identifies the condition. The sub-sequence number allows you to enter multiple conditions for a single component element.

Condition

The sequence number (**Seq**) together with the sub sequence (**Sub Seq**) number uniquely identifies the condition. Typically, the sequence number is the same as the component element within the composite to which the condition applies, although this is not mandatory. The sub-sequence number allows you to enter multiple conditions for a single component element.

The blank columns are for the sequence numbers of the elements to which the conditions apply. They must be the same as the sequence numbers in the elements detail area of the Composite window.

The condition can be one of 5 types. Valid entries are as follows:

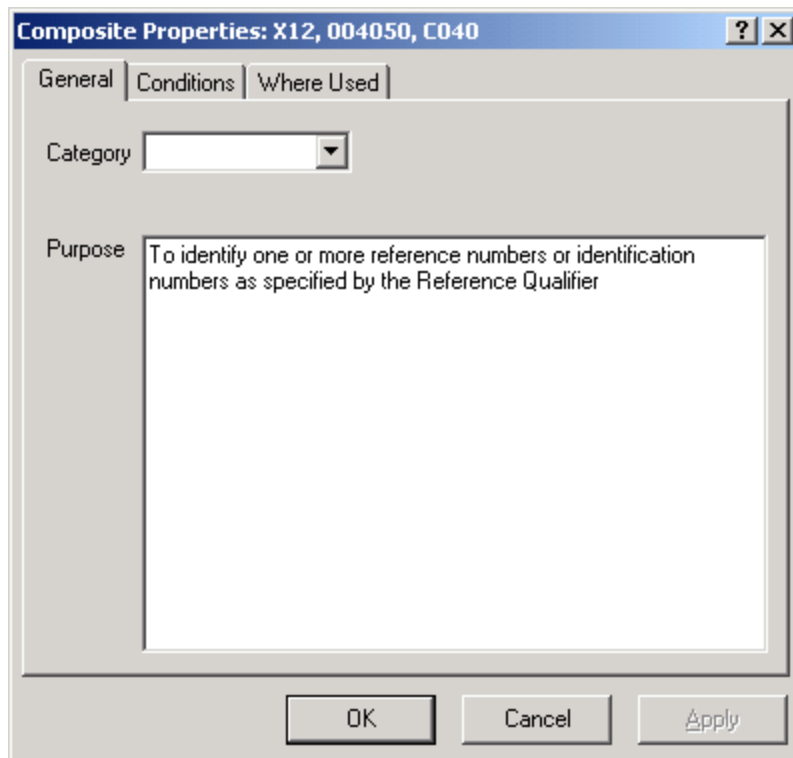
P	Paired Condition	If any data element specified is present, then all must be present.
R	Required Condition	At least one of the data elements specified must be present.
E	Exclusion Condition	Not more than one of the data elements specified may be present.
C	Conditional Condition	If the first data element specified is present, then all others must be present.
L	List Conditional Condition	If the first data element specified is present, then at least one of the others must be present.

Element Sequence Numbers

The blank columns are for the sequence numbers of the elements to which the conditions apply. They must be the same as the sequence numbers in the elements detail area of the Composite window.

(Composite Properties) General

The **General** tab allows you to select a category for a composite, which will override a higher category designation. The page also provides a place where you can write important information.



General Tab (Composite Properties Window)

Category

The category identifies the parsing and generation routines that will be used for the composite, if different from that of the segment. When you identify a category at this level, it applies to all component elements within this composite, if you do not override it with another category at a lower level. The options available for composites are as follows:

Type	Description
Delimited	Delimited routines use delimiters to parse the data, and generate all data entities separated with delimiters, unless otherwise noted. The delimiters used are the ones defined for the wrapper set that is associated with the interchange at the time of translation.

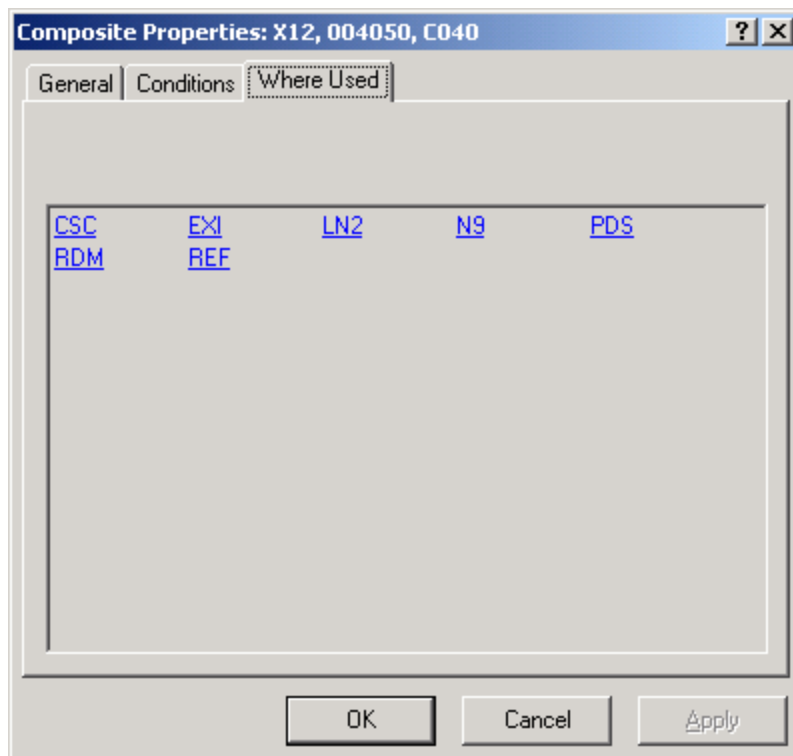
Type	Description
Fixed	Fixed routines parse the data based on the length of fields. They generate the data by padding to the full length of the field.

Purpose

The purpose field gives you up to 240 characters of space to describe the purpose of the segment or any other pertinent notes. This field is optional.

(Composite Properties) Where Used

The Where Used tab indicates which segments use this composite. To jump to a particular segment definition, double-click the segment ID.



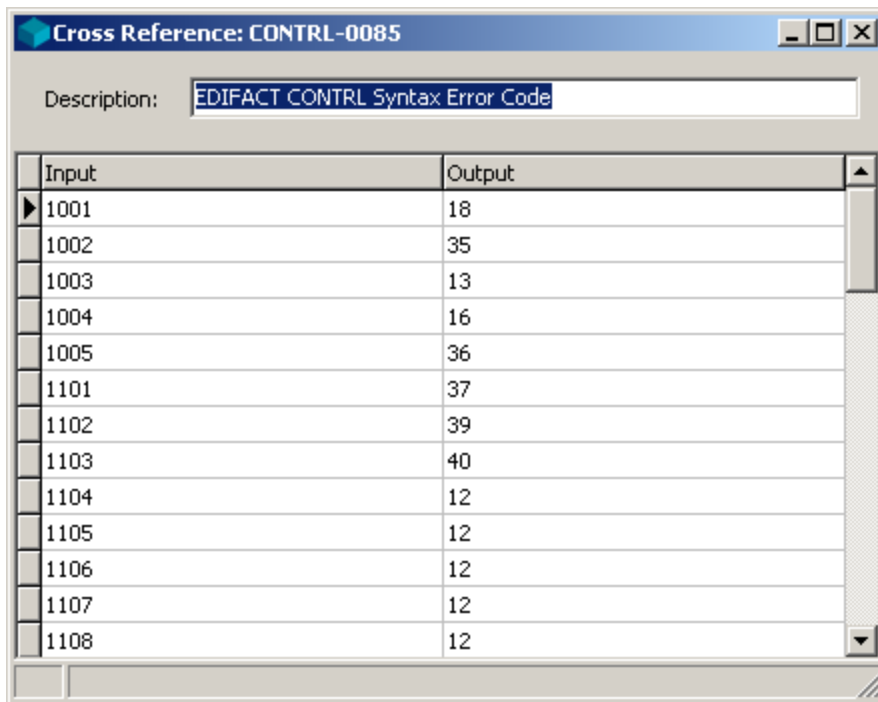
Where Used Tab (Composite Properties)

Where Used (Composite Properties)

The list box identifies all the segments that are currently using this composite. This information is maintained by the system and displays automatically based on your configurations. To jump to the definition of a particular segment displayed in the **Where Used** box, simply double-click its name.

Cross Reference Window

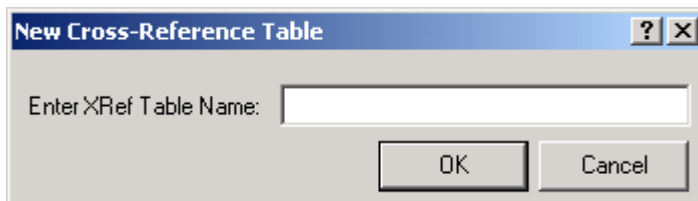
The Cross Reference window allows you to create tables that can be used to convert values in the **Input** column to values in the **Output** column or vice versa using a reverse lookup. You can define multiple tables for your system. You access a specific cross-reference table during validation or mapping using the Edibasic functions **Xref\$** or **XrefR\$**. A typical use for such a table might be to convert internal part numbers to external part numbers. The TRM uses cross-reference tables to convert internal error codes to and from external error codes.



Input	Output
1001	18
1002	35
1003	13
1004	16
1005	36
1101	37
1102	39
1103	40
1104	12
1105	12
1106	12
1107	12
1108	12

Cross Reference Window

To add a new table, right-click the list of tables in the right pane of Data Explorer and select **Add**. When the **New Cross-Reference Table** dialog box appears, you enter a name from 1 to 32 characters.



New Cross-Reference Table Dialog Box

Description

The **Description** box allows you to enter information to help you identify the purpose of this table. It can contain 1 to 80 alphanumeric characters. This field is optional.

Input

The **Input** field contains one side of the cross-reference. When you use the **Xref\$** function within Edibasic, the value in this field must match the input data. When you use the **XrefR\$** function within Edibasic, the value in this field will be the output.

Output

The **Output** field contains the other side of the cross-reference. When you use the **Xref\$** function within Edibasic, the value in this field becomes the output. When you use the **XrefR\$** function within Edibasic, the value in this field must match the input data.

Procedures (Cross Reference)

The following procedures apply to cross-reference tables, which allow you to substitute one value for another.

To Add a New Cross-Reference Table

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Cross References**.
- 2 Select **File>Add**, which will display the **New Cross-Reference Table** dialog box.
- 3 Enter the name of your new table.
- 4 Select **OK**. This takes you to a blank Cross Reference window, so you can now enter input and output values.

To Modify an Existing Cross-Reference Table

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Cross References**.
- 2 From among the list of composite elements in the right pane, double-click the name of the table you want to modify.
- 3 Make changes and close the window. The Workbench automatically saves the information.

To Save a Cross-Reference Table from Cross-Reference Table Maintenance


The Workbench automatically saves records when you exit the window. You can also make an interim save by selecting **Save** from the **File** menu.

To Delete a Cross-Reference Table

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Cross References**.
- 2 From among the list of composite elements in the right pane, select the name of the table you want to delete.



- 3 Select the **Delete** button  from the toolbar.
- 4 A **Confirm** dialog box appears requesting that you confirm or cancel the delete request. To confirm the delete request, select the **OK** button. To cancel the delete request, select the **Cancel** button.

Document Window

The Document window allows you to define the structure of a document, which is called a transaction set in X12 and a message in EDIFACT. The title bar displays the standard, version and document ID of the current document. The detail information identifies the segments and loops/groups within the document. Note that the Workbench provides visual identifiers for loops/groups. The outermost are the lines the furthest to the right. Some of the detail information, such as description, which is shaded, comes from the Segment window, and you cannot enter or change this information here. From here, you may jump to the Segment window by double-clicking a segment.

L/S	Level	Area	Seq	Tag	Description	Rqmt	Max Occ
S	0	1	10	ST	Transaction Set Header	M	1
S	0	1	20	AK1	Functional Group Response Header	M	1
L	0	1	30	AK2	Loop	✓ O	999999
S	1	1	30	AK2	Transaction Set Response Header	M	1
L	1	1	40	AK3	Loop	O	999999
S	2	1	40	AK3	Data Segment Note	M	1
S	2	1	50	AK4	Data Element Note	O	99
S	1	1	60	AK5	Transaction Set Response Trailer	M	1
S	0	1	70	AK9	Functional Group Response Trailer	M	1
S	0	1	80	SE	Transaction Set Trailer	M	1

Document Window

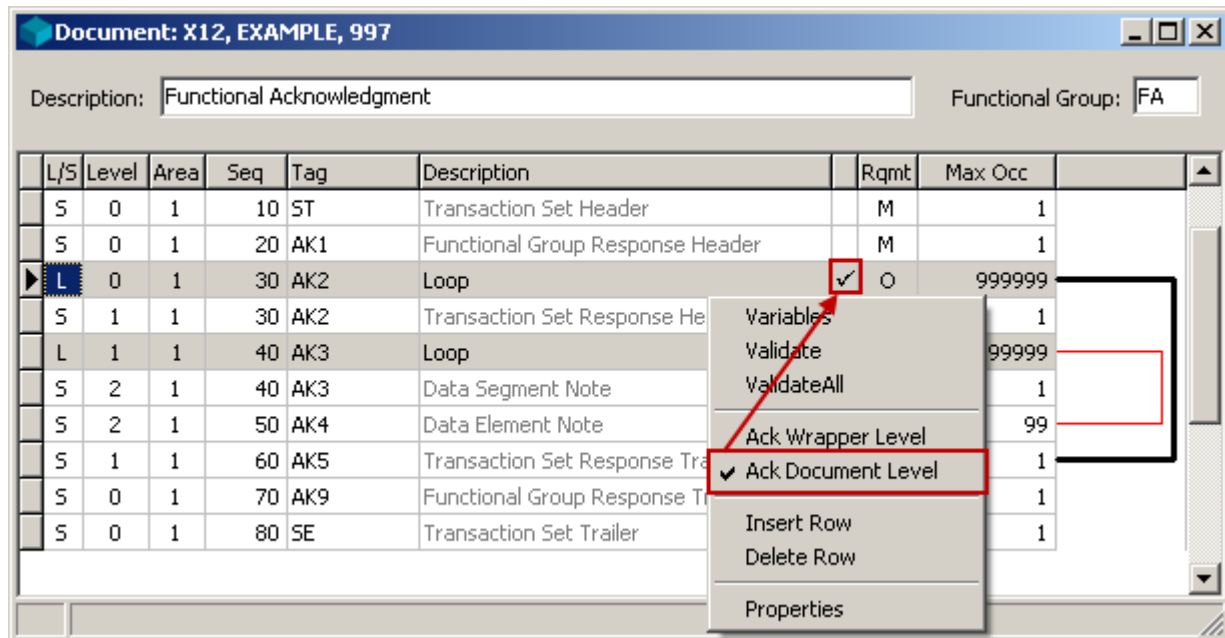
You may also right-click the window to display a list of commands. Depending on whether you click in the header or detail area and depending on the type of document, different commands appear. For example, the **Ack Wrapper Level** and **Ack Document Level** commands appear only when the document type on the Document Properties window is **Acknowledgment**. The header area is where the **Description** appears. The detail area is anywhere in the list box below the header area. Use these commands as follows:

Command	Window Area	Use
Variables	Header	To declare variables that will have a scope of the entire document, such as a global variable

Command	Window Area	Use
	Detail	To declare variables that will have a scope of the entity on which it is defined, such as a loop/group or segment.
Validate	Header	To add Edibasic code that is executed after the entire document is parsed
	Detail	To add Edibasic code that is executed when a non-repeating segment is parsed.
ValidateAll	Detail	To add Edibasic code that is executed when a loop/group or repeating segment is parsed
Ack Wrapper Level	Detail	For reconciliation purposes of an acknowledgment only, to indicate the repeating loop or segment that corresponds to a wrapper level that is more detailed than the acknowledgment itself. For example, when the acknowledgment is level 1 (acknowledges an interchange), then the Ack Wrapper Level flag indicates which repeating loop/segment acknowledges functional groups.
Ack Document Level	Detail	For reconciliation purposes of an acknowledgment only, to indicate the repeating loop or segment that corresponds to documents, that is, which repeating segment or loop/group acknowledges individual documents.
Insert Row	Detail	To insert a blank row above the selected row.
Delete Row	Detail	To delete the selected row.
Properties	Header or Detail	To open the Document Properties window.

IMPORTANT: When both the **Ack Wrapper Level** and **Ack Document Level** flags are used, they must be nested. The **Ack Wrapper Level** flag must be defined for a non-nested loop and the **Ack Document Level** flag must be defined for a loop/group or segment directly nested in the loop with the **Ack Wrapper Level** flag.

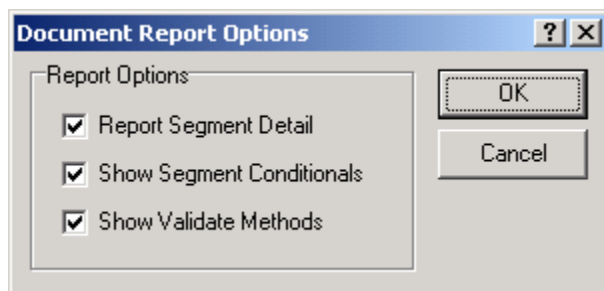
When the **blank** column contains a check, either an Edibasic method exists or an acknowledgment flag is set for a loop or segment for reconciliation.



Document Window (blank column)

The Document window has an associated properties window. You access the Document Properties window from the Document window by selecting **Properties** from the **File** menu.

You can print a document report that will show you varying degrees of information. If you select **Print** or **Print Preview** for a document, the **Document Report Options** dialog box appears.



Document Report Options Dialog Box

Report Segment Detail (Document Report Options Dialog Box)

Select the **Report Segment Detail** option to print a list of elements for all the segments. When you do not select this option, the Workbench prints a list of only segments; it will not include elements in the report.

Show Segment Conditionals (Document Report Options Dialog Box)

Select the **Show Segment Conditionals** option to include in the report a list of conditions for elements within the segment.

Show Validate Methods (Document Report Options Dialog Box)

Select the **Show Validate Methods** option to include on the report any Edibasic methods you may have written for the document, segments, composites, or elements.

Document Name

The document name, together with the standard and version, must be unique for all document definitions. It can contain any combination of 1 to 8 alphanumeric characters. You enter the name in the **New Document** dialog box when you add a new document. The standard, version and document names appear on the window title bar.



New Document Dialog Box

Description (Document)

The description field allows you to enter a textual description up to 80 characters long for the document.

Functional Group

The functional group field is optional. It can contain 1 to 8 alphanumeric characters. It is possible to assign more than one document to a functional group ID.

L/S

The **L/S** field indicates whether this entity is a loop/group (L) or a segment (S). You can choose from a list by clicking slowly twice in this field. Before you define the first segment of a loop, you must define a loop level using **L**. If you are defining a segment, within a loop or not, you would enter **S**. The loop/segment identifier, together with the area and sequence number, must uniquely identify the detail entity within the document.

Level

The level field indicates at what level the entity functions. The level can be from 0 through 32, but they must be consecutive. If an entity represents a loop, the level indicates the nesting level of the loop. If it is not within any other loop, it has a level of 0. If it is contained within one other loop, it has a level of 1, and so forth. Segments within a loop will have a level of one greater than the loop to which they belong.

The Workbench uses these levels to draw the visual aid lines for the loops, so if you make a mistake, you should notice it immediately.

Area

The area indicates to which of the 9 possible areas the segment belongs. The three most commonly used areas are as follows:

- 1 Header
- 2 Detail
- 3 Summary

The header area typically contains information about the document as a whole, such as purchase order number or name and address that applies to the entire document. The detail area typically contains item information, such as line items on a purchase order. The summary area usually contains information that summarizes the document information, such as totals.

The area becomes more important when you have a very long document, where segments occur at the same level, but in different areas. The area, together with the loop/segment identifier and the sequence number must uniquely identify the segment within the document.

Seq

The sequence number is unique for every segment in the document. The sequence number can be from 1 to 2,147,483,647 characters (theoretical limit). There may be gaps in the numbering sequence, which makes it easier to insert definitions at a later time. The sequence number of the loop entity should be the same as the sequence number of the first segment of the loop. The sequence, together with the loop/segment identifier and the area identify the detail entity within the document.

Tag

The tag is used as the segment or loop ID, depending on the entity. All segments must have tags. The tag is used during mapping to access the appropriate segment within the document. It can be 1 to 8 alphanumeric characters.

Description (Segment)

The **Description** field is brought forward from the Segment file.

Blank Column

This narrow column with a blank header contains a check mark to indicate:

- Edibasic code is attached to this entity for validation
- For acknowledgments only, a repeating entity is marked specifying a document or wrapper level for purposes of reconciliation of detail information

This allows you to see at a glance, for which entities you have written code or set an acknowledgment flag. You may right-click to use the commands from the list.

The commands, **Ack Wrapper Level** and **Ack Document Level**, are required to reconcile lower, detail levels of previous outbound messages with inbound acknowledgments. You do not need to set the flags if you only want to reconcile information at the highest level, such as at the interchange level for an interchange acknowledgment or at the functional group level for a functional group level acknowledgment.

You must set these flags to reconcile details for functional groups and documents. Specifically, you must have **Ack Wrapper Level** set in order to reconcile functional groups for an interchange-level acknowledgment, such as CONTRL with functional groups. You must also have the **Ack Document Level** set in order to reconcile documents within interchanges for an interchange-level acknowledgment, such as CONTRL with or without functional groups. You must have the **Ack Document Level** set in order to reconcile documents for a functional group-level acknowledgment, such as 997.

For example, the **Ack Document Level** flag is used for the X12 997, which acknowledges messages at the functional group level, 2. The flag is set on the AK2 loop for the Transaction Set (document), to indicate that reconciliation information is available for documents within the functional group.

Document: X12, 005010, 997
 Description: Functional Acknowledgment Functional Group: FA

	L/S	Level	Area	Seq	Tag	Description	Rqmt	Max Occ
	S	0	1	100	ST	Transaction Set Header	M	1
	S	0	1	200	AK1	Functional Group Response Header	M	1
	L	0	1	300	AK2	Loop	O	>1
	S	1	1	300	AK2	Transaction Set Respon		1
	L	1	1	400	AK3	Loop		>1
	S	2	1	400	AK3	Data Segment Note		1
	S	2	1	500	AK4	Data Element Note		99
	S	1	1	600	AK5	Transaction Set Respon		1
	S	0	1	700	AK9	Functional Group Respo		1
	S	0	1	800	SE	Transaction Set Trailer		1

Context menu options: Variables, Validate, ValidateAll, Ack Wrapper Level, **Ack Document Level**, Insert_Row

When both **Ack Wrapper Level** and **Ack Document Level** flags are used, they must be nested. The **Ack Wrapper Level** flag must be assigned to a loop that contains the **Ack Document Level** flag. For example, both the **Ack Wrapper Level** and **Ack Document Level** flags are used for the EDIFACT CONTRL, which acknowledges messages at the interchange level, 1. There are actually two types of loops: one with functional groups, which begins with the UCF loop, and one without functional groups, which begins with the independent UCM loop.

For the first type, the **Ack Wrapper Level** flag is set on the UCF loop for a functional group, should it occur.

Document: EDIFACT, 2, CONTRL

Description: Functional Acknowledgement Functional Group:

L/S	Level	Area	Seq	Tag	Description	Rqmt	Max Occ
S	0	1	10	UNH	Message Header	M	1
S	0	1	20	UCI	Interchange Acknowledgement segment	M	1
L	0	1	25	UCF	Loop	C	999999
S	1	1	30	UCF	Functional Group Response	M	1
L	1	1	35	UCM	Loop	C	999999
S	2	1	40	UCM	Message Response	M	1
L	2	1	45	UCS	Loop	C	999
S	3	1	50	UCS	Segment Error Indication	M	1
S	3	1	60	UCD	Data Element Error Indication	M	99

Context menu for UCF Loop (Seq 25):

- Variables
- Validate
- ValidateAll
- Ack Wrapper Level
- Ack Document Level
- Insert Row
- Delete Row

The **Ack Document Level** flag is set on the UCM loop for messages, which occurs within the UCF loop.

Document: EDIFACT, 2, CONTRL

Description: Functional Acknowledgement Functional Group:

L/S	Level	Area	Seq	Tag	Description	Rqmt	Max Occ
S	0	1	10	UNH	Message Header	M	1
S	0	1	20	UCI	Interchange Acknowledgement segment	M	1
L	0	1	25	UCF	Loop	C	999999
S	1	1	30	UCF	Functional Group Response	M	1
L	1	1	35	UCM	Loop	C	999999
S	2	1	40	UCM	Message Response	M	1
L	2	1	45	UCS	Loop	C	999
S	3	1	50	UCS	Segment Error Indication	M	1
S	3	1	60	UCD	Data Element Error Indication	M	99
L	0	1	270	UCM	Loop	C	999999
S	1	1	270	UCM	Message Response	M	1
L	1	1	275	UCS	Loop	C	999
S	2	1	280	UCS	Segment Error Indication	M	1

Context menu for UCM Loop (Seq 35):

- Variables
- Validate
- ValidateAll
- Ack Wrapper Level
- Ack Document Level
- Insert Row
- Delete Row

Note that another UCM loop exists below and independent of the UCF loop, which is used when there are no functional groups in the interchange. When this second loop is used, the **Ack Document Level** flag is set on the second UCM loop.

Document: EDIFACT, 2, CONTRL

Description: Functional Acknowledgement Functional Group:

L/S	Level	Area	Seq	Tag	Description	Rqmt	Max Occ
S	0	1	10	UNH	Message Header	M	1
S	0	1	20	UCI	Interchange Acknowledgement segment	M	1
L	0	1	25	UCF	Loop	✓ C	999999
S	1	1	30	UCF	Functional Group Response	M	1
L	1	1	35	UCM	Loop	✓ C	999999
S	2	1	40	UCM	Message Response	M	1
L	2	1	45	UCS	Loop	C	999
S	3	1	50	UCS	Segment Error Indication	M	1
S	3	1	60	UCD	Data Element Error Indication	M	99
L	0	1	270	UCM	Loop	✓ C	999999
S	1	1	270	UCM	Message Response		1
L	1	1	275	UCS	Loop		999
S	2	1	280	UCS	Segment Error Indication		1
S	2	1	290	UCD	Data Element Error Indication		99
S	0	1	300	UNT	Message Trailer		1

Context menu for the selected row (L 0 1 270 UCM Loop):

- Variables
- Validate
- ValidateAll
- Ack Wrapper Level
- ✓ Ack Document Level**

Rqmt

The requirement designator identifies whether the segment or loop/group is mandatory (**M**), optional (**O**), conditional (**C**), floating (**F**), or bounded (**B**). If you identify an entity as mandatory, compliance checking generates an error if it does not exist. Optional or conditional entities may or may not be present. Notice that the use of optional and conditional designators varies from standard to standard. Floating is used only for segments and is a legacy of X12. It identifies segments that are identified in one location but can theoretically appear anywhere in the document. Bounded is also used in X12, to indicate specific segments that define the boundary of a loop. The **LS** segment signals that the next segment is the beginning segment of a loop. The **LE** segment signals that the previously started loop has ended. Most loops in X12 are not bounded loops. The TRM always assumes that a segment begins a loop if, as part of its definition, it has the sequence number of the segment that ends the loop, whether it is bounded or not. The bounded designation allows the TRM to generate properly bounded loops as required, and not generate unnecessary **LS** and **LE** segments when not required.

NOTE: The floating designation used by X12, specifically for the NTE segment, was eliminated beginning in version 003060 where the NTE became optional.

Max Occ

This value identifies the maximum number of times a loop/group or segment can repeat. The TRM checks this value for compliance during validation. If the number of occurrences of the loop or segment in the incoming data is greater than this value, The TRM generates a compliance error. You can access specific occurrences within mapping using Edibasic.

TIP: To enter greater than one (>1), you can enter zero (**0**) and the Workbench automatically converts this to >1.

Procedures (Document)

You can perform these procedures from the Document window.

To Add a Document to a Standard Version

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.
- 2 From the **File** menu, select **Add**.
The **New Document** dialog box appears.
- 3 Enter the name of your new document, and select **OK**.
A blank Document window appears.

- 4 Enter a *description* (on page 562) for your document, and add segments to the document. For instructions, refer to the topic, *To Add a Loop or Segment to a Document* (on page 570).
 - 5 After you have added segments to your document, do one of the following:
 - Close the window to save the information
 - or –
 - To add another document:
 - Press **F6**
 - or –
 - Select **Add** from the **File** menu.
- The **New Document** dialog box re-appears and you can repeat steps 3-5.

To Modify an Existing Document

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.
- 2 From among the list of documents in the right pane, double-click the name of the document you want to modify.
- 3 Make changes and close the window.
The Workbench automatically saves the information.

To Save a Document

The Workbench automatically saves records when you exit the window. You can also make an interim save by selecting **Save** from the **File** menu.

To Delete a Document

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.
- 2 From among the list of documents in the right pane, select the name of the document you want to delete.



- 3 Select the **Delete** button from the toolbar.

A **Confirm** dialog box appears requesting that you confirm or cancel the delete request.

To Sequentially Display Definitions of Documents

Sometimes you want to review the list of documents. This procedure will display all definitions from all standards. The standard and version will change when you display definitions from a different standard. From the Document window, to page backward and forward within the definitions:

- 1 From the **View** menu, select **Next** or press **F7** to see the next document definition.
- 2 From the **View** menu, select **Prior** or press **F8** to see the previous document definition.

To Add a Loop or Segment to a Document

Make sure the Data Explorer window is *open* (on page 532).

- 1 If you are already in the Document window, proceed to step 2.
– otherwise –

From Data Explorer, select the document to which you want to add a loop or segment.


- 2 Place your cursor in the segments detail area.
- 3 Create a blank line.

IMPORTANT: You may insert a line anywhere in the document. They are listed in order based on the unique sequence numbers you use. To insert segments, you must have a gap in your sequence numbers. If you have no gap in your sequence numbers, you must manually resequence other segments to create a gap.

- To add a line at the end, place your cursor in the last line and press the **Tab** key or the down-arrow key.
– or –
 - To insert a line before another line, place your cursor on the detail line, and
 - Press the **Insert** key
– or –
 - Right-click, and select **Insert Row** from the menu
- 4 In the blank line, enter the appropriate information, which may include creating a new segment or loop.

NOTE: When you move your cursor to a different line, the Workbench will display the segments in the correct order.

- a) In the **L/S** (on page 563) column, type **L**(oop) or **S**(egment).
- b) In the **Level** (on page 563) column, type the level number.
- c) In the **Area** (on page 563) column, type the number of the area.
- d) In the **Seq** (on page 564) column, type the sequence number of the segment.
- e) To add a new loop or segment:

1. In the **Tag** (on page 564) column, type the segment tag, press **Tab**, and then double-click quickly.
 2. When the Segment window appears, type the appropriate information for the element, and select **OK**.
- f) To select an existing segment:
1. In the **Tag** (on page 564) column, right-click once.
 2. Click the selection button, , when it appears.
 3. From the **Select Segment** dialog box, select the appropriate element and click **OK**.
- g) In the **Rqmt** (on page 568) column, enter or select the requirement designator.
- h) In the **Max Occ** (on page 568) column, enter the maximum number of times the segment may occur.
- 5 Close the window to save your changes.

To Modify an Existing Loop or Segment in a Document

- 1 Place your cursor in the detail area, using the **Tab** key or point-and-click the mouse.
- 2 If you are not already positioned at the desired field, use the **Tab** key or the arrow keys to position the cursor to the field where you want to make the change. Type in the appropriate information.


To Delete a Loop or Segment From a Document

- 1 Place your cursor in the detail area, using the **Tab** key or point-and-click the mouse.
- 2 If you are not already positioned at the desired loop or segment, use the **Tab** key or the arrow keys to position the cursor to the line that you want to delete. Press the **CTL+DEL** keys simultaneously.
- 3 When the **Confirm** dialog box appears, select **OK** if you want to delete the item, or **Cancel** if you do not.


To Print a Document Report

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.
- 2 In the right pane, select the document you want to print.
- 3 From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file

- The **Print Preview** button  to print reports to the screen

- The **Print to PDF** button  to print to a PDF file
A **Document Report** dialog box appears.
- 4 The default report prints segment and loop information for the document. For other options, do one of the following:
 - a) To print element information on the report, check **Report Segment Details**.
 - b) To print segment conditions information on the report, check **Show Segment Conditions**.
 - c) To print the validation methods, check **Show Validate Methods**.
 - 5 Select **OK**.

To Generate a Document Text File

When you generate a document, the Workbench includes the definitions of all of its subordinate entities, such as segments and elements in the text file. Therefore, whenever you make changes to the database, you must regenerate the text file, so that you also have the latest changes available for translations. When the Workbench generates a document text file, it creates a backup first if a text file of the same name already exists.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.
- 2 In the right pane, select the document you want to generate.
- 3 Select the name of displayed document from the **Generate** menu, which will create the text file required during processing.

When you select **All**, the Workbench will generate all documents. If you do not generate a text file, this information will be available in the database, but not available for translation.

To Add Edibasic Validate Routines to Segments within Documents

You can add your own validation routines to segments within documents. Although the TRM does some compliance checking, you may want to perform additional validations. The TRM will execute the routine during parsing only when this segment occurs within this document. The validation routine may override another validation routine written for this segment from the Segment window. You should always review validation routines by printing a document report, so you will know which ones will take effect.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the desired document.
- 3 Place your cursor in the segment detail area and select the segment to which you will attach the validation routine.
- 4 Right-click and select **Validate**.

The Edit window will appear with **Method Validate** header and trailer code.

- 5 Enter your validation routine between the header and trailer code, which typically includes using the **Exception** function.
- 6 *Check your syntax for errors* (on page 585).
- 7 Close the window.

When you return to the Document window, a check mark appears, indicating user code exists for the segment.

- 8 *Print a document report* (on page 571) to review the validation routines that may be invoked for the document.

To Add Edibasic Validate Routines to Documents

You can add your own validate routines to documents. Although the TRM does some compliance checking, you may want to perform additional validations. The TRM will execute the routine during parsing whenever it uses this document definition.

Make sure the Data Explorer window is *open* (on page 532).

To add user validation routines to documents, proceed as follows:

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the desired document.
- 3 Place your cursor in the Document window header area (top part of the window).
- 4 Right-click and select the **Validate** option.

The Edibasic Edit window will appear with **Method Validate** header and trailer code.

- 5 Enter your validate routine between the header and trailer code, which typically includes using the **Exception** function.
- 6 *Check your syntax for errors* (on page 585).
- 7 *Print a document report* (on page 571) to review the validate routines that may be invoked for the document.

To Declare Variables for Validate Routines on Documents

You can declare local variables or global variables to use within validation routines for documents. You can declare local variables on the **Variables** tab or within the method on the **Validate** tab. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the **Variables** tab only. They have a lifetime from the point of declaration to the end of processing for the input stream. This allows you to store information based on validation routines during the parsing process and use it later to create a proprietary acknowledgment, for example.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.

2 In the right pane, double-click the document for which you want to create the variable.

3 Place your cursor in the window header area (top part of the window).

4 Right-click and select **Variables**.

The Edibasic Edit window appears with the **Variables** tab selected.

5 Enter your local or global variable declaration statement here.

The following example declares a global variable:

```
Global Gmyvar as Integer
```

The following example declares a local variable:

```
Dim Lmyvar as Integer
```

6 *Check your syntax for errors* (on page 585).

7 *Print a document report* (on page 571) to review the validation routines that may be invoked for the document.

To Declare Variables for Validate Routines on Segments in Documents

You can declare local variables or global variables to use within validation routines for segments within a specific document. You can declare local variables on the **Variables** tab or within the method on the **Validate** tab. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the **Variables** tab only. They have a lifetime from the point of declaration to the end of processing of the input stream. For example, you can store information based on validation routines during the parsing process and use it later to create a proprietary acknowledgment.

Make sure the Data Explorer window is *open* (on page 532).

1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.

2 In the right pane, double-click the document for which you want to create the variable.

3 Position your cursor in the segment detail area and select the appropriate segment.

4 Right-click and select the **Variables** option.

The **Variables** tab of the Edibasic Edit window appears.

5 Enter your local or global variable declaration statement here.

The following example declares a global variable:

```
Global Gmyvar as Integer
```

The following example declares a local variable:

```
Dim Lmyvar as Integer
```

6 *Check your syntax for errors* (on page 585).

7 Close the window.

When you return to the Document window, a check mark appears, indicating user code exists for the segment.

- 8 **Print a document report** (on page 571) to review the validation routines that may be invoked for the document.

Document Properties Window

The Document Properties window allows you to identify a category, if you want to override the category set at the standard version level. The category specifies parsing and generation routines. This window also allows you to identify a document type and acknowledgment level to control acknowledgment processing, and to specify whether or not to validate element codes, assuming the same flag is not already set for the entire standard version.

You access the Document Properties window from the Document window by selecting **Properties** from



the **File** menu or the **Properties** button from the toolbar.

Procedures (Document Properties)

The following procedures affect the behavior of documents.

To Specify a Category for a Document

Categories allow you to specify which routine to use for parsing or generating a document. The category is taken from that specified for the standard version, unless you specifically override it for the document. This procedure overrides the category set for the standard version.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** within the appropriate version within a standard.
- 2 In the right pane, double-click the document.



- 3 From the toolbar, select the **Properties** button.
- 4 From the Category drop-down list box select the parse and generate method you want to use for this document.

The category appears in the box.

- 5 Select **OK**.

To Specify the Document Type

The document type specifies whether to return an acknowledgment to respond to this document. There will only be an acknowledgment returned if the type selected is **Simple Document**. This does not determine if a particular instance of this document will require an acknowledgment, which is specified in the partnership definitions.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** within the appropriate version within a standard.
- 2 In the right pane, double-click the document.



- 3 From the toolbar, select the **Properties** button.
- 4 Within the **Document Type** group, select the appropriate document type.
- 5 Select **OK**.

To Check Element ID Codes for a Document

You can invoke element code validation at the standard, document, or wrapper levels. If codes are already being checked at the standard version level, the status of this document check box has no effect.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** within the appropriate version within a standard.
- 2 In the right pane, double-click the document whose codes you want to validate.



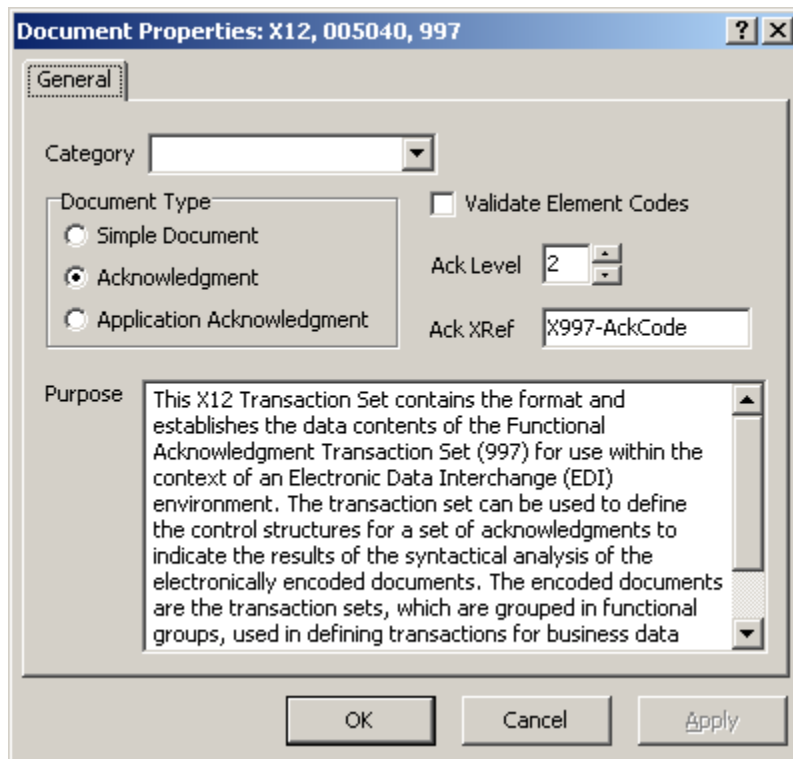
- 3 From the toolbar, select the **Properties** button.
- 4 Check **Validate Element Codes**.
A check mark appears.
- 5 Select **OK**.

(Document Properties) General

The **General** tab of the document properties window allows you to specify a document type, which controls document processing. It allows you to specify a parsing and generation category, which will override the category set at the wrapper. You may also specify if you want to validate elements for this document. You may write information pertinent to this document.

It also allows you to specify different parameters for different types of acknowledgment documents, as follows:

- **Acknowledgment:** you may also specify an acknowledgment level that will respond to a document and a cross-reference file to convert error codes from internal to external or external to internal values.
- **Application Acknowledgment:** since this type of acknowledgment is always a document level acknowledgment, you may only specify a cross-reference file to convert error codes from internal to external or external to internal values.



General Tab (Document Properties Window)

Category

The category identifies the parsing and generation routines that will be used for the document. You can specify routines at various levels in the structure. When you identify a schema at this level, it will be applied to all lower entities within this standard, if you do not override it with another schema at a lower level. It also overrides any schema set at a higher level. When you do not specify a schema here, the TRM uses the schema from the next higher level, which is the standard. The options available for the document level are as follows:

Type	Description
Delimited	Delimited routines use delimiters to parse the data, and generate all data entities separated with delimiters, unless otherwise noted. The delimiters used are the ones defined for the wrapper set that is associated with the interchange at the time of translation.
Fixed	Fixed routines parse the data based on the length of fields. They generate the data by padding to the full length of the field.
SWIFT	Supports SWIFT syntax for FIN User-to-User messages for SWIFT Block 4. This structure allows for looping and repeating segments, which is similar to the behavior of fixed and delimited categories. If the standard is identified as SWIFT, then this behavior is the default and need not be explicitly selected. This category implies SWIFT category processing for segments and elements.
XML	Uses XML syntax to parse or generate the document.

Document Type

The document type determines whether or not the TRM should ever respond to this document with an acknowledgment. Anything that is not supposed to receive an acknowledgment itself should be classified as an acknowledgment. Acknowledgments are never generated in response to other acknowledgments.

Type	Effect
Simple Document	The TRM may create an acknowledgment if so configured. Example: 850 or ORDERS
Acknowledgment	The TRM will never create an acknowledgment to this type of document. Example: 997 or CONTRL.
Application Acknowledgment	The TRM may create an acknowledgment if so configured. An example would be an X12 855, Purchase Order Acknowledgment, that is used to respond to the business information in an 850, Purchase Order. The 855 will still be able to receive a regular acknowledgment, a 997.

Validate Element Codes

When the Validate Element Codes box is not checked on the Standard Properties window, selecting this check box will allow the TRM to check values of element codes on any elements in the document for which codes are defined. When the match fails, the TRM will reject the document. If the Validate Element Codes box is checked on the Standard Properties window for this standard, checking this box has no effect.

Ack Level

This field is visible only when the document type is set to **Acknowledgment**. The **Ack Level** field identifies the level of wrapper at which the acknowledgment responds.

There will be one acknowledgment for each occurrence of that level wrapper. The options are as follows:

Level	Description
0	Acknowledgment level not applicable. This is used for documents that do not receive acknowledgments themselves, and are therefore described as acknowledgments, but that in fact provide some other function, such as AUTACK for EDIFACT.
1	Interchange level acknowledgments are in response to all functional groups, if any, and all documents that the interchange contains. There will be one acknowledgment for each interchange level wrapper.
2	Functional group level acknowledgments are in response to all documents within a functional group. There will be one acknowledgment for each functional group level wrapper.

Purpose

The **Purpose** field gives you up to 240 characters of space to describe the purpose of the document or any other pertinent notes. This field is optional.

Ack XRef

The acknowledgment cross-reference field allows you to link a cross-reference table of error codes to this acknowledgment. You would create the definition in the **Cross References** folder.

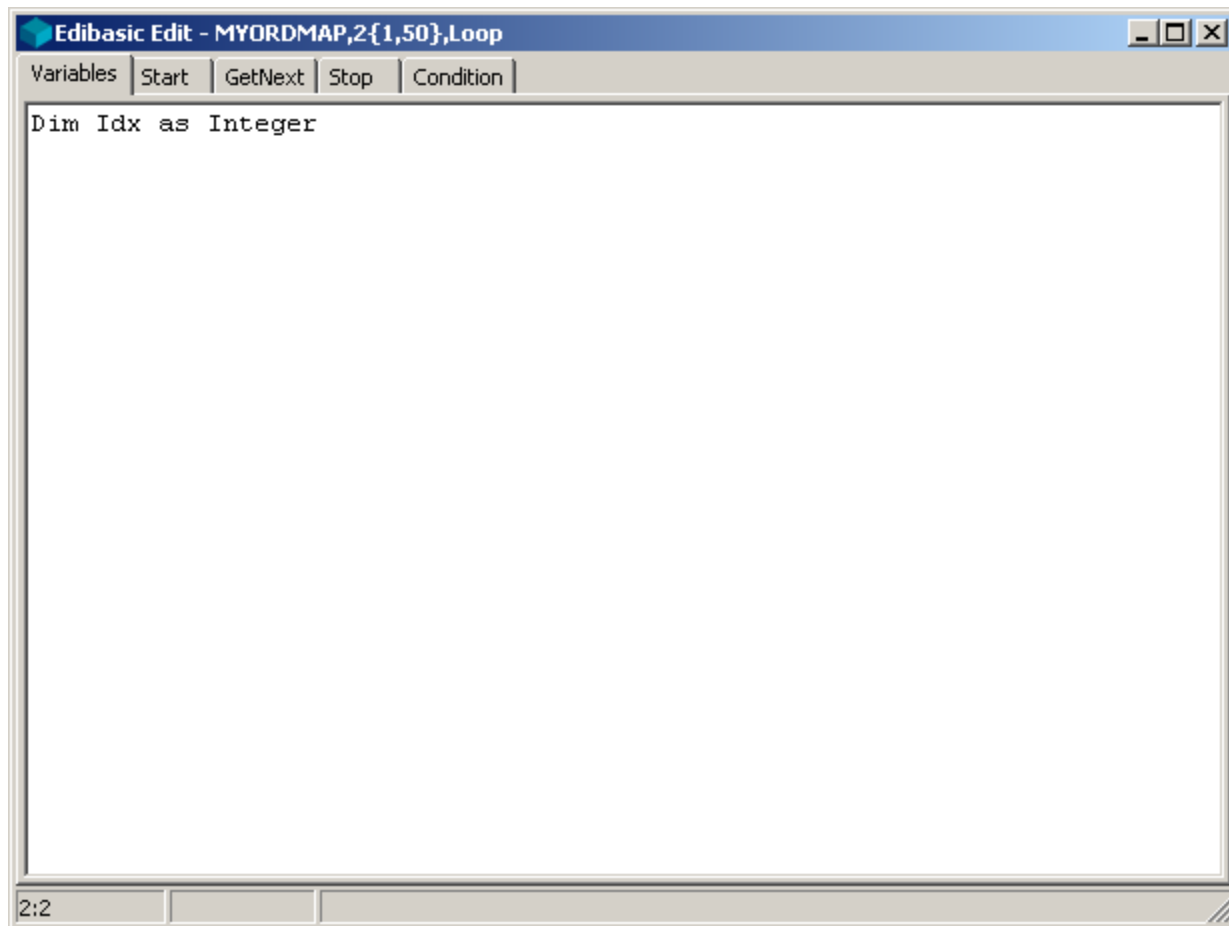
Edibasic Edit Window

To write Edibasic mapping code, you must access the Edibasic Edit window. You access the Edibasic Edit window in two ways:

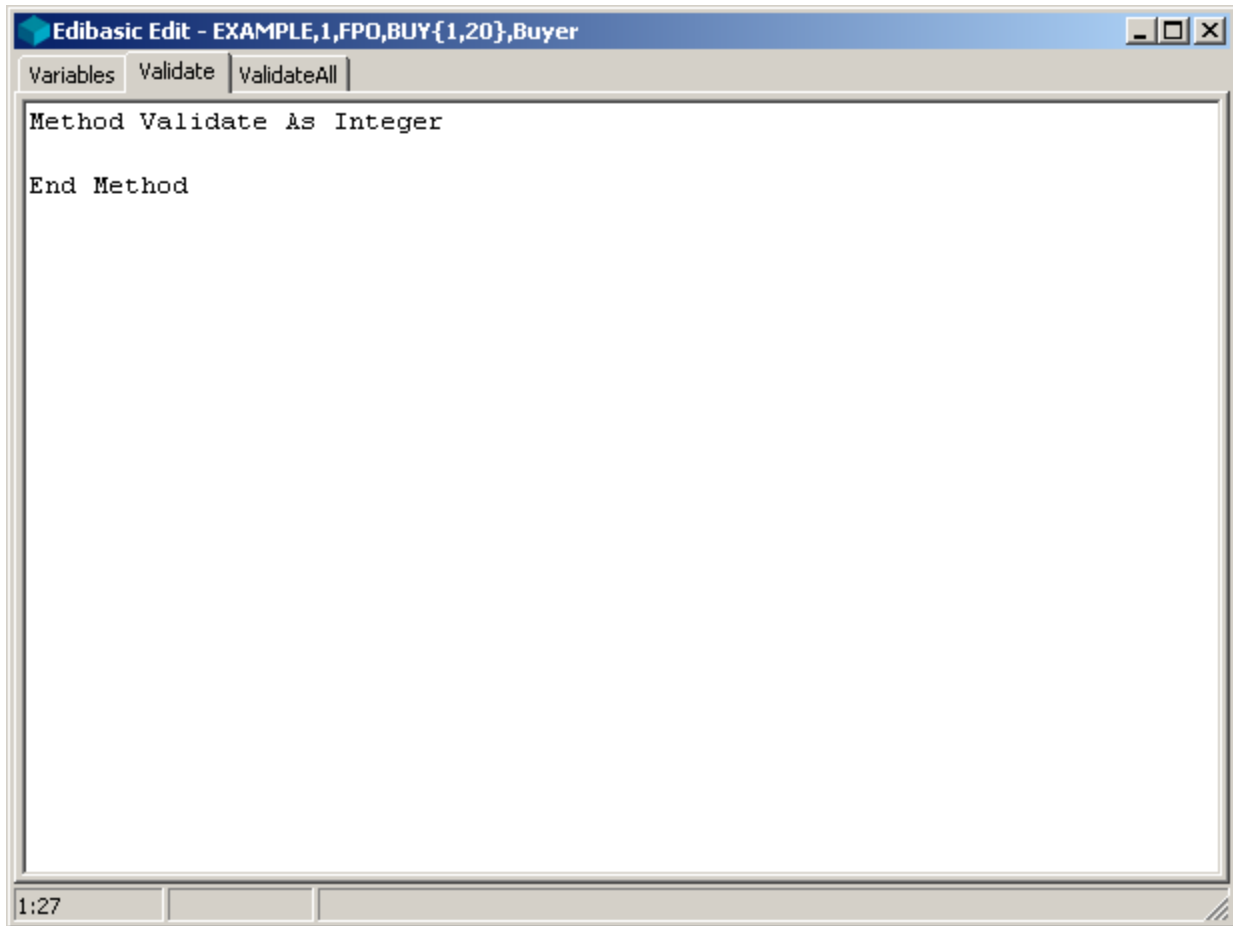
- From the Map window for mapping
 - or –
- From the standard configurations windows (Document, Wrapper, Segment, Composite, Element) to perform user validation during parsing.

For more information about Edibasic mapping techniques, refer to the topic *Advanced Mapping Techniques* (on page 185).

NOTE: There is currently a 32K limit on the size of compiled code for a single method. If you find that you need to exceed this limit, you can create user exits to access user-defined subroutines and functions. For more information on user exits, refer to *MW Translator User Exits Programming Manual*.



Edibasic Edit Window for Mapping



Edibasic Edit Window for Validation Routines on Definitions

Variables (Edibasic Edit)

The **Variables** tab allows you to create variables that have varying scopes based on the entities and levels for which they are defined. Refer to the Glossary for a definition of scope. For the effect of scope when you declare variables, refer to the discussions of the **Dim (on page 718)** and **Global (on page 721)** statements.

When you declare variables, you should always initialize them with **Method Start**. The following table describes the effect of variables declared at different levels.

Level	Typical Use
Document or wrapper	To declare variables whose scope is valid for the entire document.

Level	Typical Use
Loop or repeating segment, composite or simple element	To declare variables whose scope is valid for all occurrences. Repeating composites or simple elements may only occur in a delimited standard.
Non-repeating segment, composite element or simple element	To declare variables whose scope is valid only for that segment or composite element.

Start (Edibasic Edit)

The **Start** tab allows you to write instructions for **Method Start**. These instructions execute once before any other instructions for that entity. If the instructions are attached to the document level, any instructions you write here will execute before any other Edibasic instructions you have written for the entire document. If the instructions are attached to a specific loop, they execute before any other instructions for any of the segments or elements within the loop.

The following table describes the effect of instructions written at various levels.

Level	Typical Use
Document or wrapper	To initialize variables declared at this level. To enter Edibasic code that executes before any other code (e.g., to check a value to see if you want to generate output or not).
Loop or repeating segment, composite or simple element	To initialize variables once for all occurrences. To enter Edibasic code that executes before any other code
Non-repeating segment, composite element or simple element	To initialize variables for the segment or composite element. To enter Edibasic code that executes before any other code

GetNext (Edibasic Edit)

The **GetNext** tab allows you to write instructions for **Method GetNext**. This page only appears for repeating entities. You would use this method if you had to process loops or repeating segments differently than visual mapping, which you might find in complex nested structures where the structure of the input does not match the structure of the output. Your instructions might use conditional processing to go sequentially through the input looking for an occurrence of a loop with a specific data value, and then when properly positioned, the function **GetNext** would tell you if there were an occurrence of a repeating segment within the loop to process. If there were an occurrence of the segment, then you may map elements for the occurrence, unless a **Method Condition** indicated that you do not want to generate this instance.

Stop (Edibasic Edit)

The **Stop** tab allows you to write instructions for **Method Stop**. With **Method Stop**, you can reset the occurrences of repeating nested entities to what they were before they were initialized with **Method Start**. To do this, you use the function **ResetOcc**, which may appear on any page, but when possible, it is easier for maintenance reasons to put it on the **Stop** tab. You would write instructions here if you wanted to force Edibasic to initialize occurrences of nested loops and segments differently than visual mapping.

Condition (Edibasic Edit)

The **Condition** tab allows you to write instructions for **Method Condition**, potentially overriding the default result of mapping techniques. It is only available from a map.

Level	Typical Use
Document or wrapper	To control candidacy for generation of the document based on data content. Instructions here execute before instructions on the Start page, before routing occurs for that document and before wrappers are generated, but after TRM routing has been determined.
Loop or repeating segment, composite or simple element	To control candidacy for generation of specific occurrence of loop, segment, composite or simple element based on data content.
Non-repeating segment or composite, component or simple element	To control candidacy for generation of the segment, composite, component or simple element based on data content.

MapEle (Edibasic Edit)

The **MapEle** tab allows you to write instructions for **Method MapEle**. **Method MapEle** is the second of the two Edibasic mapping techniques, the other being **Method GetNext**, which actually causes generation of output data. When you write instructions here, you can generate an element from something other than an input element. This might be to modify the format of the element or to access a cross-reference table to replace the input value. There are many built-in functions available for these purposes. For more information on functions, refer to *Edibasic Reference* (on page 693).

Validate (Edibasic Edit)

The **Validate** tab allows you to write instructions for **Method Validate**. You access this tab by right-clicking the mouse on any entity in any of the standards definitions windows (wrapper, document, segment, element, composite). You use this method for user validation routines that will be executed during parsing. This method allows you to view the input data but not change it.

ValidateAll (Edibasic Edit)

The **ValidateAll** tab allows you to write instructions for **Method ValidateAll**. You access this tab by right-clicking the mouse on any repeating entity (loop, segment, composite element or simple element) definition on the Document, Wrapper or Segment windows. You use this method for user validation routines that will be executed during parsing. This method allows you to view the input data but not change it.

Procedures (Edibasic Edit)

These procedures are for basic syntax issues. For an introduction to Edibasic, refer to the section, *Creating Maps* (on page 155).

To Check Edibasic Syntax for Errors



- 1 After you have entered your code on a page in the Edibasic Edit window, select the **Check Errors**

button  from the toolbar.

- 2 In the lower right box, you can review the results of the error check.
 - If you have no errors, the message will indicate the number of lines checked, for example,

`1 lines compiled`.



– or –

- If you have errors, the first error will be listed with its location (line number: character) followed by text. For example, `1:10 Type name expected` indicates that the **Dim** statement on line 1, character 10 entered as `Dim I as` requires an appropriate data type, such as **Integer**.
- 3 To view additional errors, select the **Next Error** button  from the toolbar. This jumps to the next error location. If there are no further errors, this button is inactive and shaded.
 - 4 To view prior errors, select the **Prior Error** button  from the toolbar. This jumps to the previous error. If there are no previous errors, this button is inactive and shaded.

To Print Edibasic Code for Validation Routines Used during Parsing

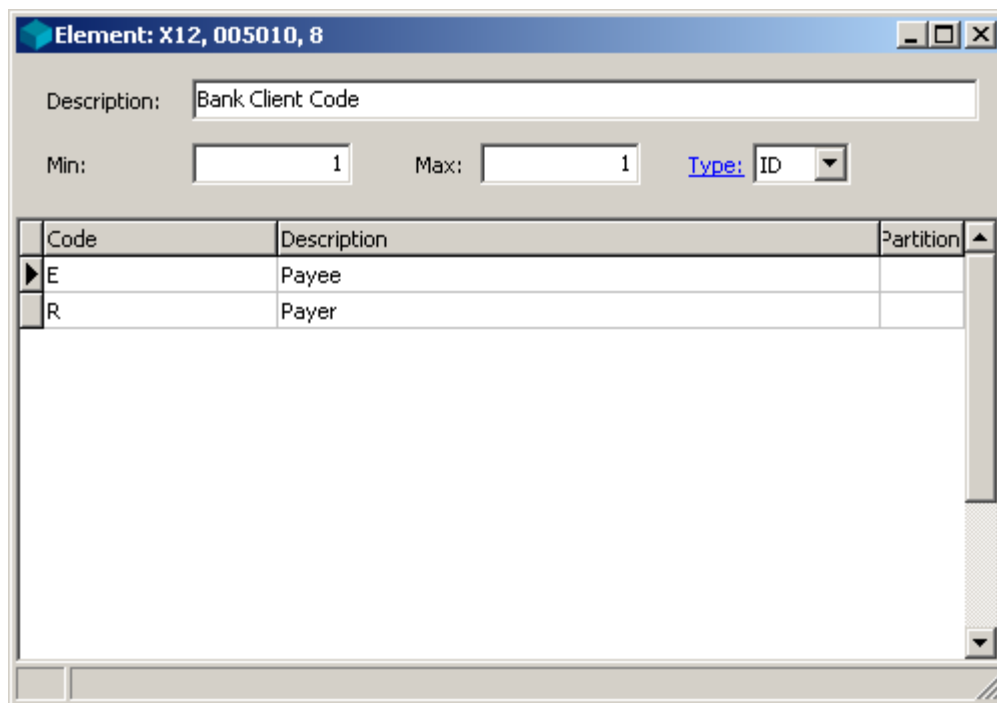
User validation routines are actually attached to entities of a standard version definition, such as a document, segment, composite, or element. In order to print the code for the validation routines, you must print a document report with the appropriate level of information.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Documents** from the expanded view of the appropriate standard and version.
- 2 In the right pane, select the document whose validation routines you want to print.
- 3 From the toolbar, select the **Print** button  or the **Print Preview** button .
- 4 A Document Report dialog box appears. The default report prints segment and loop information for the document.
 - To print element information on the report, select the **Report Segment Details** check box.
 - To print segment conditions information on the report, select the **Show Segment Conditions** check box.
 - To print the validation methods, select **Show Validate Methods** check box.
- 5 Select **OK**.
A **Report Status** box appears, indicating that the document is being printed.

Element Window

The Element window allows you to enter information for each element within a standard and version, including a list of ID codes.



Element Window

You may also right click the window to display a list of commands. Depending on whether you click in the header or detail area, different commands appear. The header area is where the Description appears. The detail area is anywhere in the list box below the header area. Use these commands as follows:

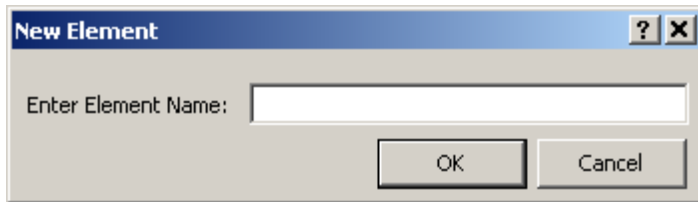
Command	Window Area	Use
Variables	Header	To declare variables that will have a scope of the element
Validate	Header	To add Edibasic code that is executed when the element is parsed
Insert Row	Detail	To insert a blank row above the selected row.
Delete Row	Detail	To delete the selected row.
Where Used	Header or Detail	To open the Where Used page of the Element Properties window.
Properties	Header or Detail	To open the Element Properties window.

This window has an associated *Element Properties window* (on page 595), which contains additional information. To access the Element Properties window, refer to the topic, *To Open a Properties Window* (on page 532).

Element Name

The element name, also called element tag and element ID, together with the standard and version uniquely identify an element. It can contain from 1 to 8 alphanumeric characters. The element name represents the reference number (X12), the element tag (EDIFACT), or some unique identifier of your choice (proprietary) for that element.

When you add an element, you specify the name in the **New Element** dialog box.



New Element Dialog Box

Description (Element)

The description field can contain from 1 to 80 alphanumeric characters. It is an optional field for you to use to describe the element.

Min

This is the minimum number of characters that the element must have, when it is used, to pass compliance checking during parsing.

Max

This is the maximum number of characters that the element may have, when it is used, to pass compliance checking during parsing.

NOTE: For Binary Coded Decimal (BCD) data, the maximum length is the number of bytes, not characters. Each byte can hold 2 numbers or one number and a sign.

To define a BCD data type, click on **Type**, and from the Standard Version window, select a base type of **Binary**, and type the value **BCD** in the Character Set column. You must also enter a format to indicate whether the data is signed or uses implied decimals. For more information about the format, refer to the function, **Format\$** (on page 737).

Type

This is the data type for this element. It may contain from 1 to 2 alphanumeric characters. It must be one of the types identified in the Standard Version window for this standard and version.

You cannot enter undefined types here. When you need to reference a new type, double-click the **Type** label to go to the Standard Version window. *After you have entered the new type* (on page 646) and closed the window, you return to the Element window, where you may select the new type.

Code

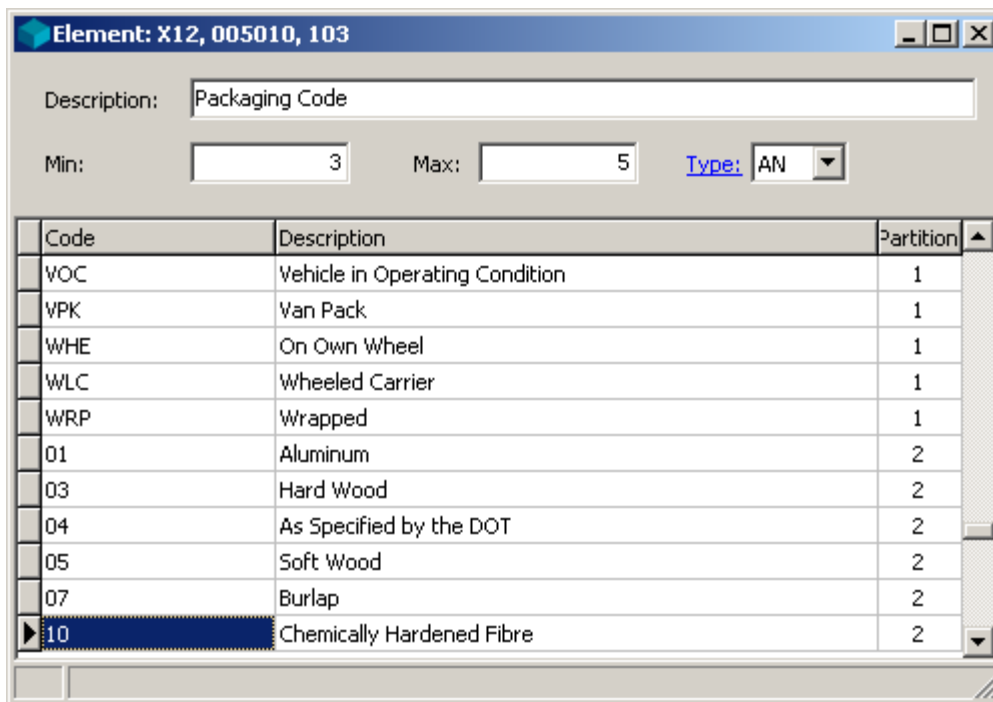
The **Code** field contains valid element codes, if any exist. When codes exist, the TRM will automatically check the incoming data to make sure it matches one of the listed codes, if you configured it to do so. To invoke automatic code validation you may check the **Validate Element Codes** box on the Standard Version Properties window, the Document Properties window, or the Wrapper Properties window for the standard, document, or wrapper to which this element belongs. When the box is checked on the Standard Version Properties window, then the check boxes on the other windows have no effect.

Description (Element Code)

The description explains the use of the particular element code. The choice of codes often depends on partnership agreement, which may have been established by a trade group.

Partition

Some elements have code lists that are identified as parts of a whole code. The code itself is divided into parts. The code list then shows all the codes associated with each of the parts. To identify which codes belong to which part of the whole code, a partition number is assigned, where 1 is for the first part, 2 is for the second part, and so on. The X12 packaging code element (103) uses the concept of a partitioned code, where it lists the possible codes for each of the partitions.



Element Window with Code Partition Values

Procedures (Element)

These procedures affect elements.

To Add an Element

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Elements** from the expanded view of the appropriate standard and version.
- 2 Select **Add** from the **File** menu.
The **New Element** dialog box appears.
- 3 Enter the **ID** (on page 588) or name of your new element. Typically, these are numeric values.

4 Select **OK**.

The Element window appears.

5 Enter appropriate values:

- a) In the **Description** (on page 588) field, type a description of the element.
- b) In the **Min** (on page 588) field, type the minimum length of the element.
- c) In the **Max (on page 589)** field, type the maximum length of the element.
- d) In the **Type** (on page 589) field, select the element type.
- e) To add a code list, refer to the topic, *To Add Element Codes for Validation* (on page 593).

6 To finish, do one of the following:

- Close the window to save the information
 - or –
- To add another element:
 - Press **F6**
 - or –
 - Select **Add** from the **File** menu.

The **New Element** dialog box re-appears and you can repeat steps 3-6.

To Modify an Existing Element

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Elements** from the expanded view of the appropriate standard and version.
- 2 From among the list of elements in the right pane, double-click the name of the element you want to modify.
- 3 Make changes and close the window. The Workbench automatically saves the information.

To Save an Element

The Workbench automatically saves records when you exit the window. You can also make an interim save by selecting **Save** from the **File** menu.

To Delete an Element

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Elements** from the expanded view of the appropriate standard and version.
- 2 From among the list of elements in the right pane, select the name of the element you want to delete.

- 3 From the **File** menu, select **Delete**, or select the **Delete** button  from the toolbar.

A **Confirm** dialog box appears requesting that you confirm or cancel the delete request.

- 4 Select **OK**.

To Sequentially Display Definitions of Elements

Sometimes you want to review the list of elements. This procedure will display all definitions from all standards. The standard and version will change when you display definitions from a different standard. From the Element window to page backward and forward within the definitions:

- 1 From the View menu, select **Next** or press **F7** to review the next element definition.
- 2 From the View menu, select **Prior** or press **F8** to review the previous element definition.

To Add User Validation Edibasic Routines to Elements

You can add your own validation routines to elements. Although the TRM does some compliance checking, you may want to perform additional validations. The validation routine may be overridden by another validation routine written for this element from a Segment window. The TRM will execute the routine during parsing whenever it encounters this element, if another definition does not override it. You should always review validation routines by printing a document report, so you will know which ones will take effect.

Make sure the Data Explorer window is *open* (on page 532).

To add user validation routines to an element, proceed as follows:

- 1 In the left pane, select **Elements** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the desired element.
- 3 Place your cursor in the Element window header area (top part of the window).
- 4 Right-click your mouse button, and select the **Validate** option.

The Edibasic Edit window appears with **Method Validate** header and trailer code.

- 5 Enter your validation routine between the header and trailer code, which typically includes using the **Exception** function.
- 6 *Check your syntax for errors* (on page 585).
- 7 *Print a document report* (on page 571) to review the validation routines that may be invoked for the document.

To Declare Variables for Elements

You can declare local variables or global variables to use within validation routines for elements. You can declare local variables on the **Variables** tab or within the method on the **Validate** tab. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the **Variables** tab only. They have a lifetime from the point of declaration to the end of processing for the input stream. This allows you to store information based on validation routines during the parsing process and use it later to create a proprietary acknowledgment, for example.

Make sure the Data Explorer window is *open* (on page 532). To create local or global variables for an element, proceed as follows:

- 1 In the left pane, select **Element** from the expanded view of the appropriate standard and version.
- 2 Double-click the element in the right pane for which you want to create the variable.
- 3 Place your cursor in the window header area (top part of the window).
- 4 Right-click your mouse button, and select the **Variables** option.

The Edibasic Edit window appears with the **Variables** tab selected.

- 5 Enter your local or global variable declaration statement here. The following example declares a global variable:

```
Global Gmyvar as Integer
```

The following example declares a local variable:

```
Dim Lmyvar as Integer
```

- 6 **Check your syntax for errors** (on page 585).
- 7 **Print a document report** (on page 571) to review the validation routines that may be invoked for the document.

To Add Element Codes for Validation

In order to validate element codes automatically, you must enter them in the Element window. You can enter codes in any order or insert codes in any place, because the Workbench automatically sorts them when it displays the information.

From within the Element window:

- 1 If you are entering codes on an existing code list, select any item in the list and press the **INSERT** key.
– otherwise –
To enter a new code list, place your cursor in the code area and proceed to step 2.
When you begin to type, the insert symbol, an asterisk (*), appears to the left of the code field.
- 2 Enter appropriate values:
 - a) In the **Code (on page 589)** column, type the code.
 - b) In the **Description (on page 589)** column, type an optional description

- c) If the code is part of a partition, such as the X12 element 103, packaging code, type the number of the partition in the **Partition (on page 590)** column.
- 3** To insert another code, press the **INSERT** key again.
– or –
To exit the maintenance window and add the new information, select the **CLOSE** button.
– or –
To exit the edit mode without saving changes, select the **ESCAPE** key.


To Delete Element Codes

From within the Element window:

- 1** To delete an element code, select the code record that you want to delete.
- 2** Right-click your mouse, and select **Delete Row**.
A **Confirm** box appears.
- 3** Select **OK**.

To Automatically Validate Element Codes

You can configure a standard version to validate element codes. The default for public standards is to not validate codes, which saves processing time. You can easily invoke validation at the standard, document, or wrapper level. You can also enter element codes for proprietary standards or add codes to public standards when partnership agreements require you to do so. If this validation is not specific enough, you can write your own Edibasic validation routines.

- 1** To validate all codes for a standard:
 - a) In the left pane of Data Explorer, expand the display for the appropriate standard and version and select **Standard Version Profile**.
 - b) In the right pane, double-click the profile.
The Standard Version window appears.

 - c) From the toolbar, select the **Properties** button.
 - d) Check **Validate Element Codes**.
- 2** To validate all codes for a document, when you have not done so already for the standard as in step 1:
 - a) In the left pane of Data Explorer, expand the display for the appropriate standard and version and select **Documents**.
 - b) In the right pane, double-click the document.
The Document window appears.

c) From the toolbar, select the **Properties** button



d) Check **Validate Element Codes**.

3 To validate all codes for a wrapper, when you have not done so already for the standard as in step 1:

a) In the left pane of Data Explorer, expand the display for the appropriate standard and version and select **Wrappers**.

b) In the right pane, double-click the document.

The Wrapper window appears.

c) From the toolbar, select the **Properties** button



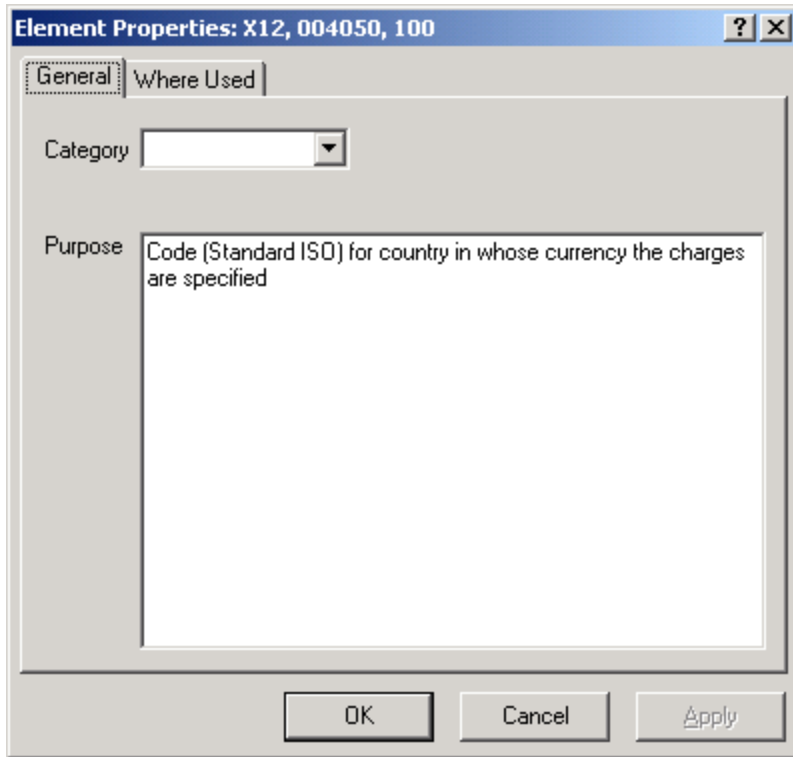
d) Check **Validate Element Codes**.

Element Properties Window

The Element Properties window allows you to identify a category to change the processing type from a higher level if needed and a purpose statement. It also provides a list of segments where this element is used.

(Element Properties) General

The **General** tab allows you to specify a parsing and generation category that will override the category set at a higher level. It also provides space for you to enter information about the element.



General Tab (Element Properties Window)

Category

The category identifies the parsing and generation routines that will be used for the element, if the routine is different from that of the composite element when the element is a component element, or if the routine is different from the segment when the element is a simple element. The options available are as follows:

Type	Description
Binary	An element that contains only binary data in octets, from binary 00000000 to 11111111. Since the length of such elements is undefined, it is preceded by another element identifying the length of the binary element.

Type	Description
Delimited	Delimited routines use delimiters to parse the data, and generate all data entities separated with delimiters, unless otherwise noted. The delimiters used are the ones defined for the wrapper set that is associated with the interchange at the time of translation.
Fixed	Fixed routines parse the data based on the length of fields. They generate the data by padding to the full length of the field.
NDDS	Non-delimited data in a delimited standard (NDDS) identifies a fixed length field in a delimited, variable-length entity. You use this option to force an otherwise variable entity to be padded to its maximum length.
SWIFT	A variable length element delimited with carriage-return and linefeed, CR-LF. The element may or may not begin with a colon (:) or a hyphen-bracket (-). This element type is only valid for SWIFT documents. Do not use this category for SWIFT wrapper elements. Defining a SWIFT category for an element used in wrappers will produce undefined results.
SWIFT Block	<p>A variable length element, surrounded by brackets, where the tag and value are separated by a colon:</p> <pre data-bbox="472 995 594 1024" style="text-align: center;">{tag:value}</pre> <p>SWIFT Block elements can occur in any order. This element category is only valid for SWIFT wrappers. Do not use this category for SWIFT document elements. Defining a SWIFT Block category for an element used in documents will produce undefined results.</p>
SWIFT Fixed	An element of a specified length, similar to regular fixed elements, except that the SWIFT fixed element may be truncated or omitted by using the right bracket (}). SWIFT Fixed elements must occur in a predefined order. This element category is only valid for SWIFT wrappers. Do not use this category for SWIFT document elements. Defining a SWIFT Fixed category for an element used in documents will produce undefined results.
Variable	<p>Variable routines are the same as fixed routines, in that there are no delimiters. The variable routine, however, does not pad the output data. The segment may have variable length fields. For this to be useful, you must explicitly add delimiter fields or length fields as elements to your document definitions. You would use this if you needed to create quoted, comma-delimited data, where commas might also appear within quotes as in the following example:</p> <pre data-bbox="378 1640 951 1669" style="text-align: center;">SEG, "This is a text field",25,15,"Another Text field"</pre> <p>In this case, you define and map the comma delimiters, excluding the commas within quotes, as actual fields.</p>
XML	Generates XML syntax for this element when parent XML element is defined as a loop.

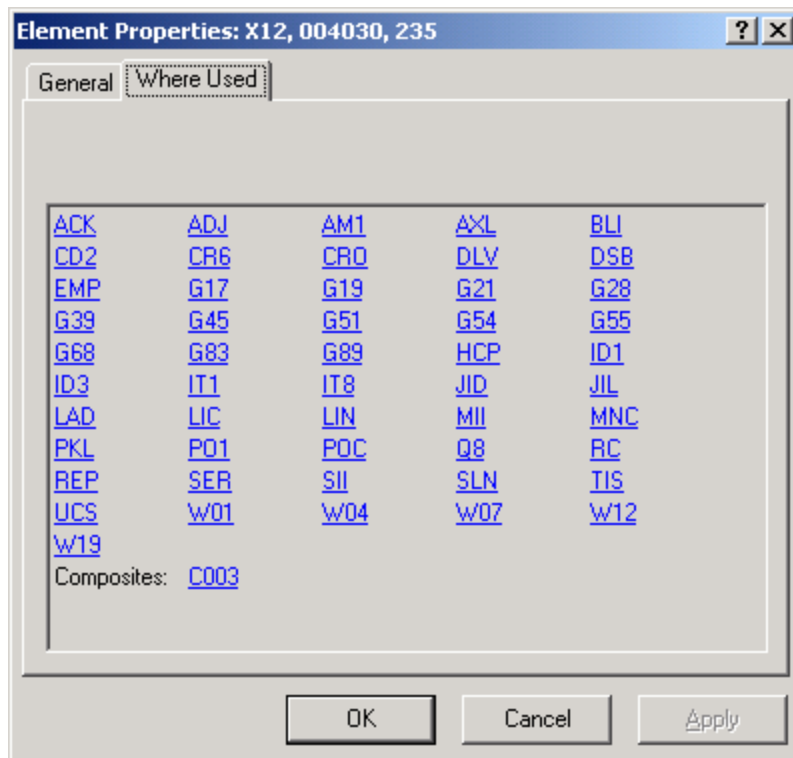
Type	Description
XML Attribut	Creates an attribute for this element using the description as the attribute name and the mapped value as the attribute value.
XML CDATA	Adds the value for this element to a CDATA section.
XML Element	Generates XML syntax for this element when the parent element is defined as a segment, which means that all elements contain no nested elements and no attributes.

Purpose

The purpose field gives you up to 240 characters of space to describe the purpose of the element or any other pertinent notes. This field is optional.

(Element Properties) Where Used

The **Where Used** tab specifies the segments and composites that use the element. Composites are listed after segments.



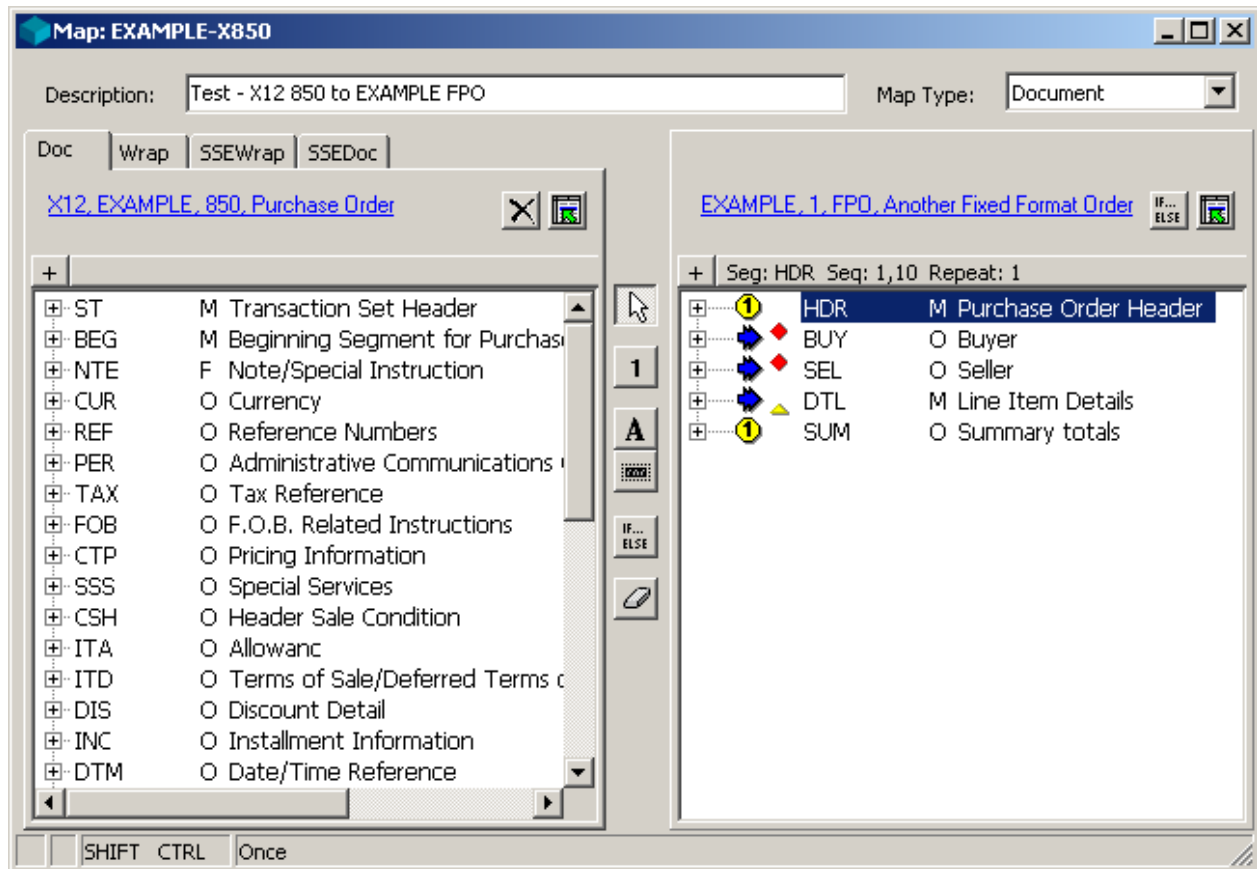
Where Used Tab (Element Properties Window)

Where Used

This field is filled by the Workbench, and indicates the composite elements or segments where this element is used. Composites are listed after segments. Notice that the values are colored and underlined indicating that you can click the composite or segment name to go immediately to that definition.

Map Window

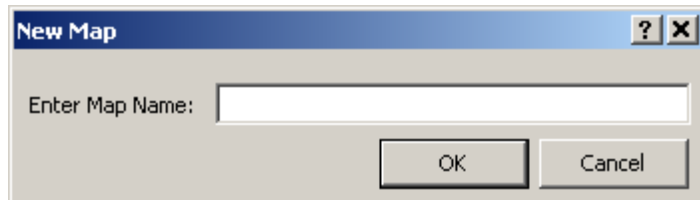
You use the Map window for both methods of mapping: visual and Edibasic. You use visual techniques from the Map window. You write Edibasic code from the Edit window. Most maps are a combination of these two techniques. Whatever you cannot do with visual techniques, you can do with Edibasic.



Map Window

Map Name

You can create a name up to 15 characters long to identify each map. Your map name might reflect the file name you create when you generate the map, or it might be some other descriptive name. You enter the map name in the **New Map** dialog box when you add a map.



New Map Dialog Box

Description

This description is additional information for you to use to identify the purpose of the map.

Map Type

Maps are used to generate outbound documents. To do so you have access to various types of information that the TRM captures: incoming document (Doc tab) and wrapper (Wrap tab) information, as well as status, statistics, and error information (SSEDOC and SSEWRAP tabs). The map type keeps you from selecting the wrong type of map for a trade agreement profile or an acknowledgment profile. The map type also allows access to information applicable for that type, because not all information is available to all types or at all times.

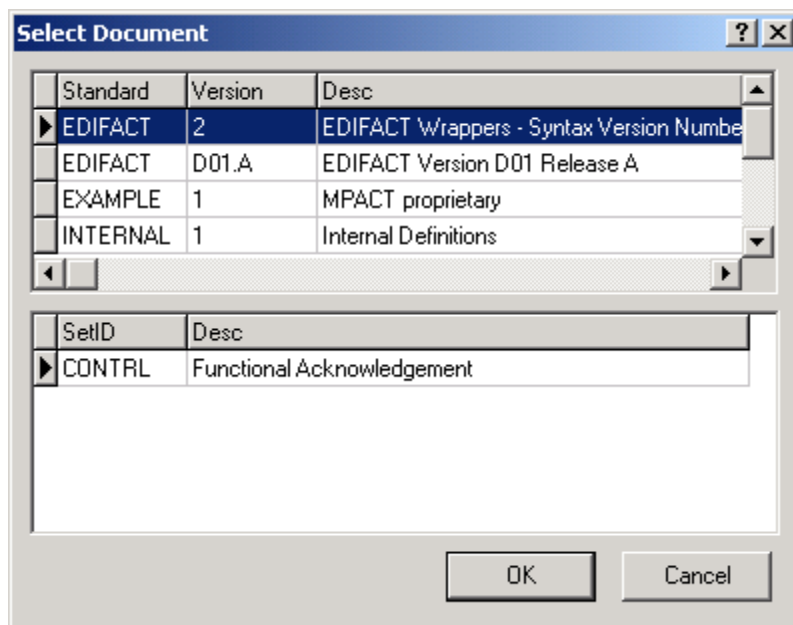
Map Type	Information Accessibility	Examples
Document	This map type creates the segments for a document. This is the only type that can map from information listed on the Doc tab.	EDIFACT ORDERS X12 850
Wrapper	This map type creates the envelope segments (wrappers). This map type typically uses partnership configuration information. The Doc tab is not accessible for this map type.	EDIFACT UNBNFG X12 ISA
Acknowledgment	This map type creates a document that responds to received data. It is also called a backward acknowledgment, because it is returned to the sender of the original document. The Doc tab is not accessible for this map type.	EDIFACT CONTRL X12 997

Map Type	Information Accessibility	Examples
Summary Document	This map type creates a document that responds to information in an interchange and is included in that interchange. Another use for a summary document is a forward acknowledgment. The Doc tab is not accessible for this map type.	EDIFACT AUTACK X12 980

Source Document


The area to the left of the window contains information about the source for the translation. Above the list box is the name of the source document, if you have chosen one. You may not have a source document defined if you are mapping acknowledgments, for example, because such information typically comes from processing information, which is stored in the SSEDOC and SSEWRAP locations. The selection

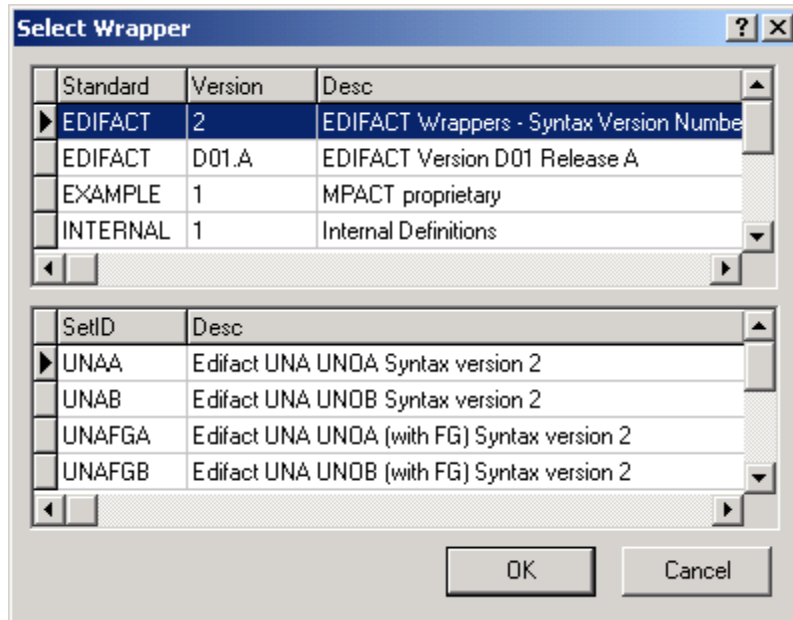
buttons to the right allow you to select a document  or to cancel the selection of a document . If you choose the **Select Source Document** button, a **Select Document** dialog box appears.



Select Document Dialog Box (Map Window)



Source Wrapper

Since you can map some items from a source wrapper, you may select a source wrapper from the **Wrap** tab using the selection button . This displays the **Select Wrapper** dialog box.

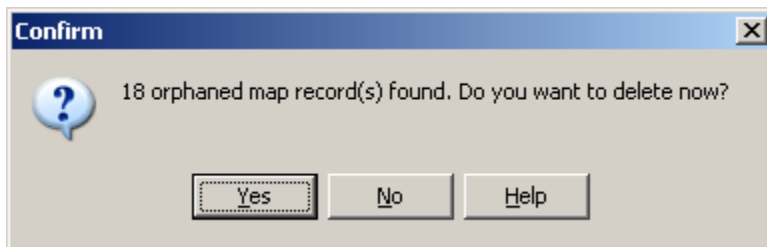


Select Wrapper Dialog Box (Map Window)

Destination Document

The area to the right of the window contains information about the destination. Above the list, the Workbench displays the destination document. The Edibasic method button  to the right allows you to enter variables and methods at the document level. The selection button  to the far right allows you to select a destination document.

If you decide to update a map, perhaps by using a more recent version of the standard, the Workbench will automatically retain mappings where the area and sequence numbers match in the old and new destination documents. If the area and sequence numbers for the items you have mapped change between the old and new documents, you will have inappropriate maps or orphaned records in your map, which are mapping references to nonexistent destination locations. Elements with invalid mappings appear as if they have not been mapped. You will receive a confirmation message that asks if you want to delete the orphaned records or not.



If you have a large number of orphaned records, check the destination document you selected to make sure you did not select the wrong document. You may decide to retain the orphaned records until you have a chance to check the mapping report. You will have an opportunity later to delete the orphaned records when you reopen the map.

Selected Item Definition Detail

The areas immediately above the list boxes display detail information about the item highlighted in the list box itself. The display varies depending on what you have selected:

Selection	Item Type	Description
Segment	Seg:	The segment ID, the sequence number (area, segment sequence number), and the maximum number of times it can repeat.
Loop(X12)/ group(EDIFACT)	Loop:	The loop/group ID, the sequence number (area, segment sequence number), and the maximum number of times it can repeat.
Simple element	Ele:	The element reference number, the element sequence number, the data type, and the minimum and maximum lengths.

Selection	Item Type	Description
Composite element	Comp:	The composite element ID and the sequence number of the composite element within the segment.
Component element	Ele:	The element reference number, the component sequence number (composite sequence number:sequence number of component within the composite), the data type, and the minimum and maximum lengths.

When you first see the items in a document, you see only the highest levels of information: loops/groups and segments that are not in loops/groups. You can immediately spot the loops/groups, because there is no description associated with them. Highlighting an item also tells you whether it is a loop/group. To display lower levels of information, double-click the item. You can also collapse a display by double-clicking on the higher-level items. When a destination item has been mapped, an icon appears next to it. Any item with subordinate items is displayed in black. The lowest level element is displayed in a contrasting color and has a dot next to it.

Mapping Source

If you are mapping from a specific source, such as, a document, a wrapper, or any of the internal information associated with the input that is stored in the data element stores, these segments will appear in the source list box to the left. Items listed in black indicate that there are additional levels of information available, which you can reveal by double-clicking on the item. Items in a contrasting color and/or preceded by a dot indicate that this is the most basic level of information available for the item.

DOC

Information on this tab is automatically displayed, and the source of the mapped item is highlighted, when you select a destination item that has been mapped from one of the items on the source side. You can also manually select this tab, and any information associated with it.

WRAP

Information on this tab is automatically displayed, and the source of the mapped item is highlighted, when you select a destination item that has been mapped from one of the items on the source side. You can also manually select this tab, which displays available information. To map from a previously parsed wrapper, you can select this tab and then select the wrapper from which you want to select items.

SSEDOC

This tab displays the locations for the status, statistics, and errors information related to the incoming document. We typically use this kind of information when we map control documents. Select the tab to display its information.

SSEWRAP

This tab displays the locations for the status, statistics, and errors information related to the incoming wrapper. Select the tab to display its information.

Mapping Destination (Map Window)


The segments associated with the output document appear in the source list box to the right. Items listed in black indicate that there are additional levels of information available, which you can reveal by double-clicking on the item. Items in a contrasting color and/or preceded by a dot indicate that this is the most basic level of information available for the item.









Once you have selected a destination item to be mapped, an icon will appear next to the item. This provides a clear visual representation of what you have and have not mapped.

When you select a new source or destination for an existing map, you can cause existing mapped items to be inappropriate for the new structure. This can cause inappropriate mapping instructions and orphaned records. To find orphaned records, you should print a map report and compare it with what is displayed in the Map window. The map window will not show the unlinked mappings, but the map report will.

Source Of Mapped Item (Map Window)

When you select a destination item that has been mapped, the source of the mapping displays at the bottom of the window and icons appear next to the mapped item. If the destination is mapped from a location in the data element store, the item from which it is mapped is also highlighted on the source side, with the detail information appearing at the top. The possible source statements that can appear at the bottom of the window and the related icons that appear next to the mapped item are as follows (two icons that have no source statement associated with them are at the end):

Source Statement	Icon	Description
Once		The selected segment, loop, composite element or simple element is to be mapped only once, regardless of the number of times the input repeats, even when there is no input.

Source Statement	Icon	Description
Depends on		The selected non-repeating entity such as a segment, loop or element is to be mapped not at all or once based on the existence of the identified source, which can be repeating or non-repeating.
Repeats from		The selected repeating entity such as a segment or loop is to be mapped based on the number of repetitions of the identified source.
Source		Identifies the source of the element. For the syntax of the input location name, refer to the topic, Data Element Names (on page 698).
Literal		The element is mapped from the literal value displayed within quotes (maximum 43 characters).
Field		The element is mapped from one of the internal fields, a list of which is available from the Select Field dialog box.
Method		The item is mapped using Edibasic MapEle or GetNext methods.
		The item has Edibasic code on a Variables , Start or Stop tab.
		The item has Edibasic code on a Condition tab.



Procedures (Map)

These procedures provide some basic instructions for mapping tasks. For more information about mapping, refer to the topic, *Creating Maps* (on page 155).

To Enter a New Map


Make sure the Data Explorer window is *open* (on page 532).



- 1 In the left pane, select **Maps**.
- 2 From the **File** menu, select **Add**.
The **New Map** dialog box appears.
- 3 Enter the name of your new map.
- 4 Select **OK**.
A blank Map window appears.
- 5 Enter a description for the map.
- 6 Select a map type: wrapper, document, acknowledgment, or summary document. Once a map is saved as a wrapper map, you may not switch to another type.

- 7 To map from information in a document or wrapper, on the left side of the detail area, select the tab that describes the type of input information, the Wrap tab for a wrapper or the Doc tab for a document.
- 8 Click the **Select Source** button, , and select the appropriate source definition.
- 9 On the right side of the detail area, click the **Select Destination** button, , and select the appropriate destination definition.
- 10 Enter mapping instructions using visual techniques such as drag-and-drop or Edibasic techniques, and close the window. The Workbench automatically saves the information.

To Modify an Existing Map

Make sure the Data Explorer window is *open* (on page 532).

NOTE: If you have modified the definition of the source or destination document, by either adding new segments or elements or modifying their characteristics, you must reselect that document using the appropriate button . This process applies the previous mapping instructions to the new document. If the structure of the new document has changed, you may have orphaned instructions. You will be asked if you want to either delete the orphaned records or leave them for future cleanup. To find orphaned mappings, print a map report and compare it with the mappings visible in the Map window.

- 1 In the left pane, select **Maps**.
- 2 From among the list of maps in the right pane, double-click the name of the map you want to modify.
- 3 If you have made changes to the source document, select the **Select Source** button .
- 4 If you have made changes to the destination document, select the **Select Destination** button .
- 5 Make changes and close the window. The Workbench automatically saves the information.

To Save a Map

The Workbench automatically saves records when you exit the window. You can also make an interim save by selecting **Save** from the **File** menu.

To Delete a Map

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Maps**.
- 2 From among the list of maps in the right pane, select the name of the map you want to delete.
- 3 From the **File** menu, select **Delete**.

– or –

From the toolbar, select the **Delete** button






A **Confirm** dialog box appears.

- 4 Select **OK**.

To Print a Map Report

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Maps**.
- 2 In the right pane, select the map you want to print.
- 3 From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file
- The **Print Preview** button  to print reports to the screen
- The **Print to PDF** button  to print to a PDF file

To Generate a Map Text File

When you run tests, the TRM uses the configuration information stored in generated text files. When you make changes to a map, the changes are logged to a database. When you generate a map, the Workbench creates a text file containing the mapping instructions from the database. Therefore, whenever you make changes to the database, you must regenerate the text file, so that you also have the latest changes available for translations. When the Workbench generates a text file, it creates a backup (*.bak) first if a text file of the same name already exists.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Maps**.
- 2 In the right pane, select the map you want to generate.
- 3 From the **Generate** menu do one of the following:
 - To create a text file for a specific map, select the map.
– or –
 - To create text files for all the maps in your database environment, select **All**.

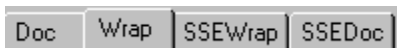
To Map an Input Wrapper Field to a Document Element Using Visual Mapping

Open a map, such as EXAMPLE-X850, which is loaded with the Workbench.

- 1 Expand the destination display (right list box) until you can see the element you want to map.

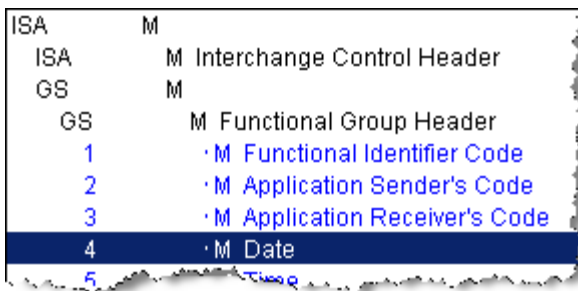


- 2 In the left box, click the **Wrap** tab at the top of the source window.



This displays a collapsed view of the input wrapper associated with the document that you are mapping.

- 3 Expand the source list until you can see the element from which you want to map.



- 4 Left-click the source element, and, holding the mouse button down, move the pointer to the destination element and release the button.
 - An icon should appear next to the mapped element.
 - If you have not already mapped the segment to which it belongs, a default mapping of **Once** is given to the segment, and an icon appears next to it.
 - When you select the item, the word **Source:** followed by the element name of the source field you selected should appear at the bottom of the window.

Source: WRAP.GS{1,20}.4

To Map a Literal to an Element Using Visual Mapping

Open a map, such as EXAMPLE-X850 that is loaded with the Workbench.

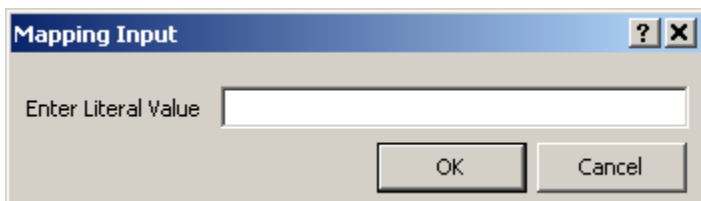
- 1 In the right box, expand the destination display until you can see the element you want to map.
- 2 You can use one of two visual techniques to map:

- Select the **Literal** button , then click the destination item.

– or –

- Click the destination item, and with the cursor still positioned over the item, right-click, and select **Literal**.

The Mapping Input dialog box appears.



- 3 Type the text in the box labeled **Enter Literal Value**, and select **OK**.

Do not enter quotation marks unless they are actually part of the literal value. This technique uses a maximum of 43 characters. If your literal is longer, use the Edibasic method MapEle.


- A check mark should appear next to the mapped element.
- If you have not already mapped the segment to which it belongs, a default mapping of **Once** is given to the segment, and a check mark appears next to it.
- When you select the item, the word **Lit:** followed by the text you entered enclosed in quotation marks should appear below the box at the left.



To Map an Internal Field to an Element Using Visual Mapping

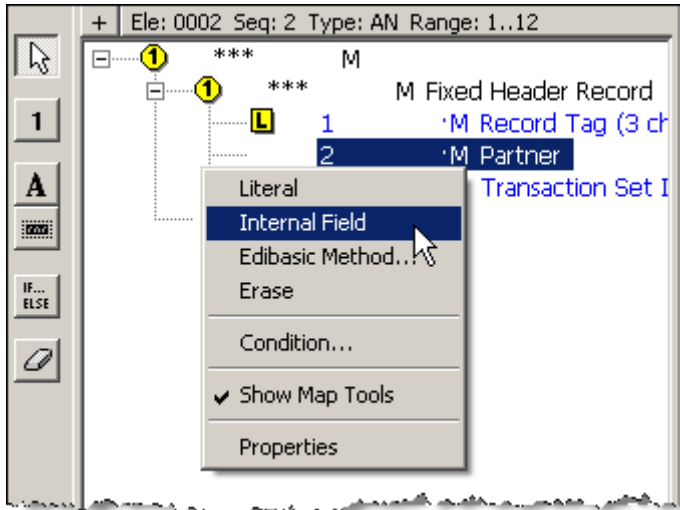
Open a map, such as EXAMPLE-X850 that is loaded with the Workbench.

- 1 In the right box, expand the destination display until you can see the element you want to map.
- 2 You can use one of two techniques to map:

- Click the **Internal Field** button , and then click the target element in the destination set.

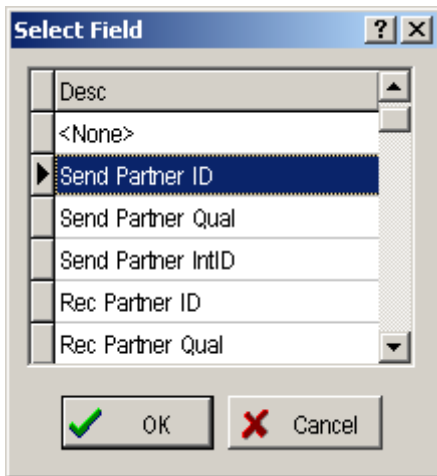
– or –

- When your cursor is directly over the target element, right-click, and select **Internal Field**.



The **Select Field** dialog box appears.

- Select the appropriate source field, and select **OK**.



- An icon should appear next to the mapped element. If you have not already mapped the segment to which it belongs, a default mapping of Once is given to the segment, and an icon appears next to it.
- When you select the item (highlighted), the word **Field:** followed by the name of the field you selected should appear at the bottom of the window.


Field: Send Partner ID

To Delete (Erase) a Mapping Instruction

Open a map, such as EXAMPLE-X850 that is loaded with the Workbench.

This technique only deletes the mapping instructions, not the item(s) to which the instructions refer.

Whenever you erase a mapping instruction, any instructions associated with subordinate items are also erased.


- 1 In the right box, expand the destination display until you can see the item whose mapping instructions you want to delete.
- 2 You can use one of two techniques to delete the instructions:
 - Click the **Erase** button  , and then click the target item in the destination set
– or –
 - Click the target item in the destination set, then when your cursor is directly over the target item, right-click and select **Erase** from the menu.
- 3 Select **OK** to confirm.


An icon next to the previously mapped item should disappear, and any icons next to subordinate items disappear

To Add Edibasic Instructions to a Map at the Document or Wrapper Level

Open a map, such as EXAMPLE-X850 that is loaded with the Workbench.

At this level, you may declare variables (**Variables** tab) whose scope includes the entire document or wrapper. You may also write Edibasic instructions in the Start method (**Start** tab) that will execute before any other instructions in the map. The Edibasic methods you may use at this level are Start and Condition.

- 1 Above the right box, select the **Document Variables and Methods** button  , which is next to the **Select Destination** button.
The Edibasic Edit window appears.
- 2 Select the **Variables** tab to declare variables or the **Start** tab to write Edibasic instructions. Select the **Condition** tab to write Edibasic code to determine whether to generate the document or wrapper.
- 3 On the **Variables** tab, declare your local and global variables.
- 4 On the **Start** tab, enter your Edibasic instructions for the Start method. If you declared variables, you should initialize those variables within the Start method.

- 5 Don't forget to check your syntax by selecting the **Check Errors** button .

IMPORTANT: Instructions on the **Condition** page execute before instructions on the **Start** page, before routing occurs for that document and before the wrapper is generated, but after TRM routing has been determined.

To Add Edibasic Instructions at the Loop/Group, Segment, or Composite Element Level


Open a map, such as EXAMPLE-X850 that is loaded with the Workbench.

At this level, you may create variables (Variables tab) whose scope includes the entire loop, repeating or non-repeating segment, or composite element. You may also write Edibasic instructions that will execute at the beginning of each occurrence of the loop, segment, or composite element. The Edibasic methods you may use at this level are Start, GetNext, Stop, and Condition.

IMPORTANT: There is a limit of 32K for any code page.

- 1 Choose one of the following techniques to access the Edit window:
 - This first technique allows you to consider alternatives before you make a choice, and subsequently limits the display of tab options within the Edit window if you choose **Conditions** or **Variables** (choosing **Edibasic Method** displays all tabs, which has the same effect as using the second technique).
 - a) Select the loop, segment, or composite element for which you wish to write instructions. Make sure if you want the loop, rather than the first segment of the loop, the information at the top of the list box says loop.
 - b) With your cursor still positioned over the loop, segment, or composite element, click the right mouse button, and select one of the three Edibasic options. Select **Variables** if all you want to do is declare and initialize variables. Select **Conditions** if all you want to do is write conditions to control processing. Select **Edibasic Method...** if you want to write instructions for more than one type of Edibasic method.

– or –

 - This second technique is the same as choosing **Edibasic Method** in the previous technique, where all tabs are displayed:
 - a) Select the **Edibasic Method** button .
 - b) Click the loop, segment, or composite element for which you wish to write a method. Make sure you have selected the correct item. If you want the loop, rather than the first segment of the loop, the information at the top of the box should say *Loop*.

The Edibasic Edit window appears.

- 2 Select the appropriate tab.

For non-repeating entities (single segments and composites), you will only be able to write Start and Condition methods. For repeating entities (loops and repeating segments), you will be able to write Start, GetNext, Stop, and Condition methods.

- 3 Write the appropriate instructions.



- 4 Check your syntax by selecting the **Check Errors** button.

- 5 Close the window.

In the Map window, an icon should appear next to the mapped item. If you have not already mapped the higher level(s) to which it belongs, a default mapping of **Once** is given to any unmapped higher level(s), and an icon appears next to it/them. When you select the item it will be highlighted, and the word Method should appear at the bottom of the destination set list box.

To Add Edibasic Instructions at the Element Level

Open a map, such as EXAMPLE-X850 that is loaded with the Workbench.

At this level, you may create variables (**Variables** tab) whose scope includes the only the element. You may also write Edibasic instructions that will affect the creation of that element. The only two methods you can use at this level, Condition and MapEle, each with its own tab.


IMPORTANT: There is a limit of 32K for any code page.

- 1 Choose one of two techniques to access the Edibasic Edit window:
 - This first technique allows you to consider alternatives before you make a choice, and subsequently limits the display of tab options within the Edit window if you choose Condition (choosing **Edibasic Method** displays both tabs, which has the same effect as using the second technique).

- a) Select the element for which you wish to write a method.
- b) With your cursor still positioned over element, right-click, and select **Condition** or **Edibasic Method**.

– or –

- This second technique is the same as choosing **Edibasic Method** in the previous technique, where all tabs are displayed:

- a) Select the **Edibasic Method** button .
- b) Click the element for which you wish to write a method.

The Edibasic Edit window appears.

- 2 Select the appropriate tab. You can write Edibasic instructions for only two methods, MapEle and Condition.
- 3 Write the instructions.



- 4 Check your syntax by selecting the **Check Errors** button.

In the Map window, a check mark should appear next to the mapped item. If you have not already mapped the higher level(s) to which it belongs, a default mapping of **Once** is given to any unmapped higher level(s), and a check mark appears next to it/them. When you select the item it will be highlighted, and the word Method should appear at the bottom of the destination set list box.

Segment Window

The Segment window allows you to define the structure of a segment. The detail information identifies the simple or composite elements within the segment. Some of the detail information, such as description, data type, and minimum and maximum lengths, actually come from the Element database, and you cannot enter this information here. From here you can go directly to the Element window by double-clicking on an element or to the Composite Element window by double-clicking a composite element.

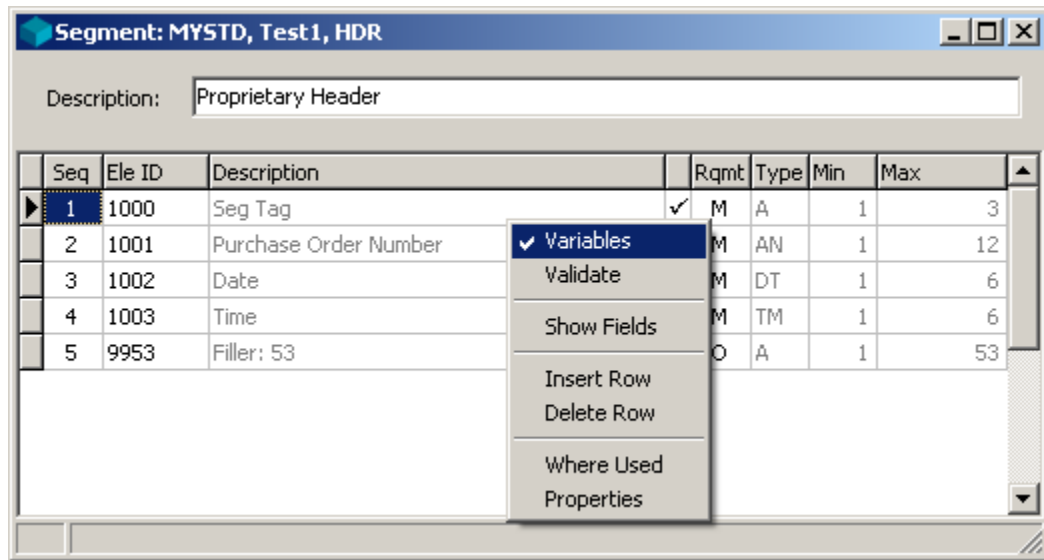
Seq	Ele ID	Description	Rqmt	Type	Min	Max
1	737	Measurement Reference ID Code	O	ID	2	2
2	738	Measurement Qualifier	O	ID	1	3
3	739	Measurement Value	C	R	1	20
4	C001	Composite Unit of Measure	C			
5	740	Range Minimum	C	R	1	20
6	741	Range Maximum	C	R	1	20
7	935	Measurement Significance Code	O	ID	2	2
8	936	Measurement Attribute Code	C	ID	2	2
9	752	Surface/Layer/Position Code	O	ID	2	2
10	1373	Measurement Method or Device	O	ID	2	4
11	1270	Code List Qualifier Code	C	ID	1	3
12	1271	Industry Code	C	AN	1	30

Segment Window (Document Segment)

You may also right-click the window to display a list of commands. Depending on whether you click in the header or detail area, different commands appear. The header area is where the Description appears. The detail area is anywhere in the list box below the header area. Use these commands as follows:

Command	Window Area	Use
Variables	Header	To declare variables that will have a scope of the entire segment or loop/group
	Detail	To declare variables that will have a scope of the entity on which it is defined, such as a composite or element.
Validate	Header	To add Edibasic code that is executed after the entire segment is parsed
	Detail	To add Edibasic code that is executed when a non-repeating composite or simple element is parsed.
ValidateAll	Detail	To add Edibasic code that is executed when a repeating composite or element is parsed
Show Fields	Detail	To display the column Fields, to associate an element with a particular internal field, such as Date.
Show Repeats	Detail	To display the column Rep, to specify the maximum number of occurrences for a repeating composite or element.
Insert Row	Detail	To insert a blank row above the selected row.
Delete Row	Detail	To delete the selected row.
Where Used	Header or Detail	To open the Where Used page of the Document Properties window.
Properties	Header or Detail	To open the Document Properties window.

When the blank column contains a check, an Edibasic method is defined for this element.



Segment Window (blank column)

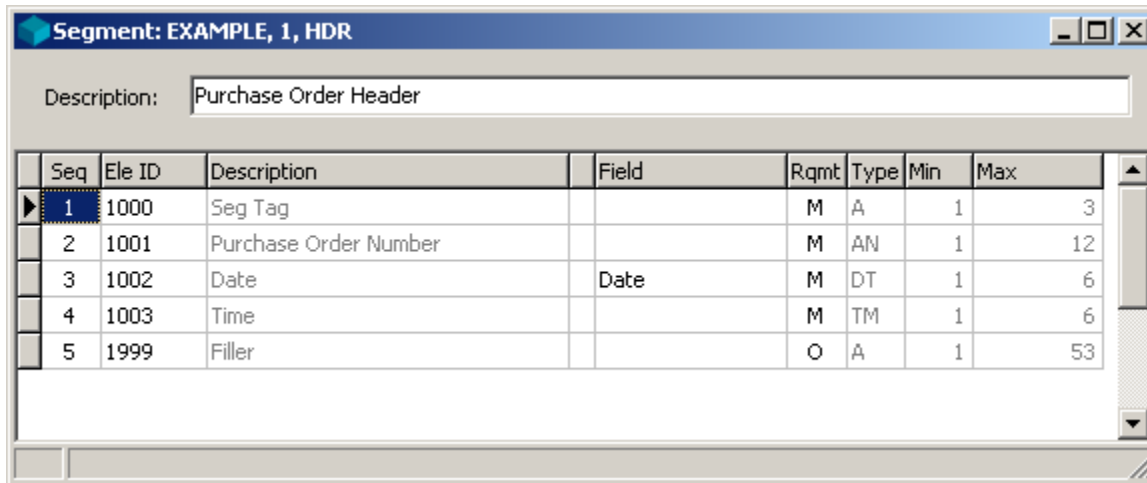
For delimited standards, composite elements and simple elements may repeat. When a definition includes repetitions greater than one, a Rep (repetitions) column also displays. You can manually display the column by right clicking and selecting **Show Repeats** from the pop-up menu.

The screenshot shows a window titled "Segment: X12, 005010, EB" with a description "Eligibility or Benefit Information". Below the description is a table with columns: Seq, Ele ID, Description, Rqmt, Rep, Type, Min, and Max. Row 3 is selected, and a context menu is open over it, with "Show Repeats" highlighted.

Seq	Ele ID	Description	Rqmt	Rep	Type	Min	Max
1	1390	Eligibility or Benefit Information Code	M		ID	1	2
2	1207	Coverage Level Code	O		ID	3	3
3	1365	Service Type Code	O	99	ID	1	2
4	1336	Insurance Type Code	O		ID	1	3
5	1204	Plan Coverage Description	O		AN	1	50
6	615	Time Period Qualifier	O		ID	1	2
7	782	Monetary Amount	O		R	1	18
8	954	Percentage as Decimal	O		R	1	10
9	673	Quantity Qualifier	C		ID	2	2
10	380	Quantity	C		R	1	15
11	1073	Yes/No Condition or Respo	O		ID	1	1
12	1073	Yes/No Condition or Respo	O		ID	1	1
13	C003	Composite Medical Procedu	O				
14	C004	Composite Diagnosis Code Pointer	O				

Segment Window (Repeat column)

You can also show the Fields column by right-clicking and selecting **Show Fields** from the pop-up menu. This column shows for all wrapper segments, but only for document segments if an internal field has been assigned to an element within the segment.

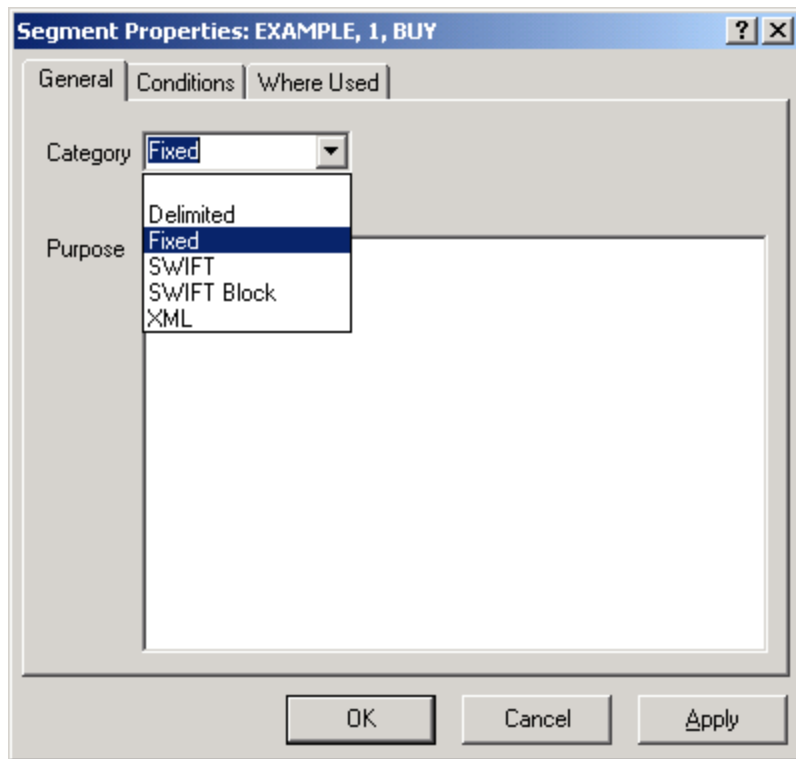


The screenshot shows a window titled "Segment: EXAMPLE, 1, HDR" with a description field containing "Purchase Order Header". Below the description is a table with the following columns: Seq, Ele ID, Description, Field, Rqmt, Type, Min, and Max. The table contains five rows of data.

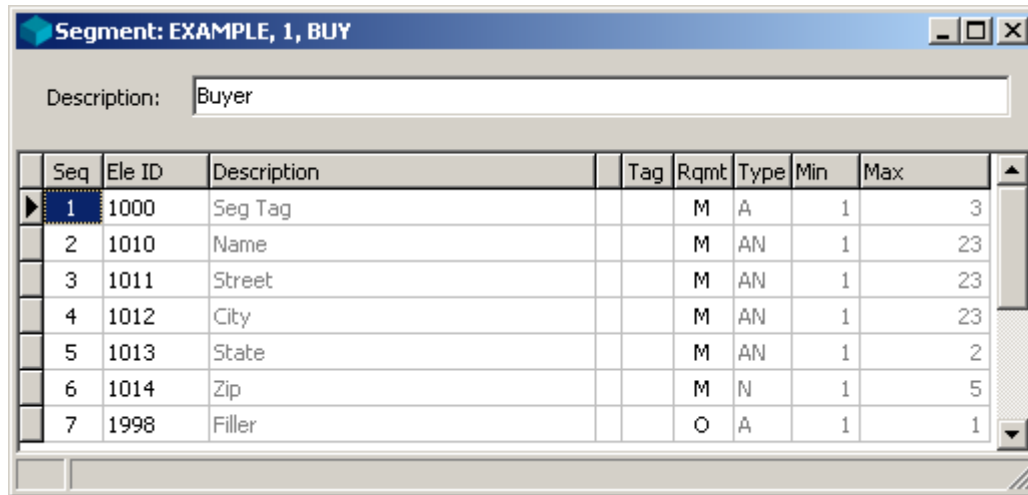
Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	1000	Seg Tag		M	A	1	3
2	1001	Purchase Order Number		M	AN	1	12
3	1002	Date	Date	M	DT	1	6
4	1003	Time		M	TM	1	6
5	1999	Filler		O	A	1	53

Segment Window (Field Column, Document Segment)

Note that the Tag column appears only when you explicitly define the segment as fixed-length; that is, you select **Fixed** in the Category box on the Segment Properties window for the segment. This is necessary to define a non-initial element as the segment tag in a fixed-length segment.



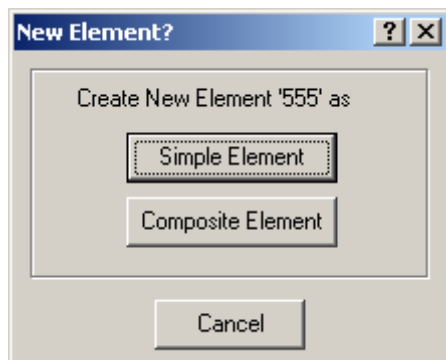
Segment Properties Window (Category Fixed)



Segment Window (Document Segment, Category Fixed)

The Segment window has an associated *Segment Properties window* (on page 633). The Segment Properties window contains information about category, purpose, conditions, and where used. To access the Segment Properties window from the Document window, from the **File** menu, select **Properties**.

From here you may go directly to the Element window by double-clicking on an element or to the Composite Element window by double-clicking on a composite element. When you create a new element from this window, double-clicking on the element displays the New Element dialog box.



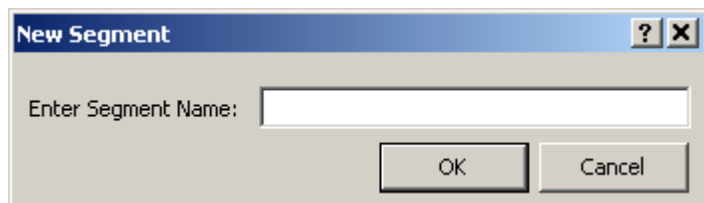
New Element Dialog Box

Segment Name

The segment name, together with the standard and version, must be unique for all segment definitions. It may contain any combination of 1 to 8 alphanumeric characters.

IMPORTANT: The Segment Name must match the actual data. For example, if the value of the segment tag in the data is **HDR**, then the Segment name must be **HDR**.

You specify the segment name or tag in the **New Segment** dialog box when you add a new segment.



New Segment Dialog Box

Description

The description field allows you to enter a textual description up to 80 characters long for the segment.

Seq

The sequence number is unique for every element in the segment. The sequence number can be from 1 to 32,767 (theoretical limit) characters. There can be sequential gaps in the numbers, but their numeric order determines the order of the elements. When you define a standard, it is good practice to leave gaps between composites or elements, so you don't have to manually resequence them later if you want to insert one. If you define conditions for this segment, these sequence numbers are used in the conditions on the **Conditions** tab of the Segment Properties window to relate the condition to the appropriate element.

Ele ID

The Ele ID, together with the standard and version uniquely identify the element or composite element. It can be 1 to 8 alphanumeric characters. For elements or composite elements this is referred to as the data element or composite element reference number in X12 and the data element or composite element tag in EDIFACT, respectively.

Description (Element)

This field is display only. It comes from and can be accessed in the Element window.

Blank Column

This narrow column with the blank header contains a check mark whenever you have attached Edibasic validation routines to this entity. This allows you to see at a glance, for which entities you have written validation routines.

Tag

This feature allows you to identify the segment tag in different locations for fixed-format segments. The TRM uses the segment tag value to identify the segments in the incoming data. For delimited segments, the tag is always at the beginning of the segment. For fixed-format segments, the tag is by default the first element of the segment. Users may change the element to be used as the tag. The **Tag** column only appears when the segment is explicitly defined as fixed-length from the Segment Properties window. This field identifies which element will be used as the segment tag. Select the tag field for an element, and press any key to place or remove a **T** in the field.

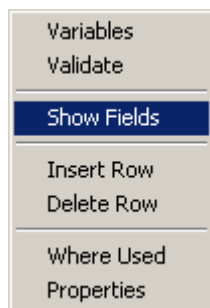
Seq	Ele ID	Description	Tag	Rqmt	Type	Min	Max
1	1000	Seg Tag		M	A	1	
2	1010	Name	T	M	AN	1	23
3	1011	Street		M	AN	1	23
4	1012	City		M	AN	1	23
5	1013	State		M	AN	1	23
6	1014	Zip		M	N	1	23
7	1998	Filler		O	A	1	1

Segment Tag in Non-initial Position for Fixed-length Segment (Segment Window)

Field

This column displays any internal fields that have been assigned to a particular element, simple or component. It is used most often for elements in wrapper segments, but may also be used for elements in document segments. When the TRM parses the incoming wrappers, the element values associated with these internal fields are used to find partner profiles and sender and recipient profiles.

The column is visible for document segments only when an element has been associated with an internal field. To display the column, right-click and select **Show Fields** from the menu.



For instructions to associate an internal field with an element, refer to the topic, *To Assign an Internal Field To an Element* (on page 630).

Note that most internal fields are useful for wrappers. During parsing, information is stored in these field locations, typically to allow the TRM to find further definitions and configurations for processing.

Segment: EXAMPLE, 1, HDR
Description: Purchase Order Header

Seq	Ele ID	Description	Field	Rqmt	Type	Min	Max
1	1000	Seg Tag		M	A	1	3
2	1001	Purchase Order Number		M	AN	1	12
3	1002	Date	Date	M	DT	1	6
4	1003	Time		M	TM	1	6
5	1999	Filler		O	A	1	53

Segment Window (Field Column, Wrapper Segment)

Rqmt

The requirement designator identifies whether the element is mandatory (**M**), optional (**O**), or conditional (**C**). If you identify an entity as mandatory, compliance checking generates an error if it does not exist. Optional or conditional entities may or may not be present. Notice that the use of optional and conditional designators varies from standard to standard.

Rep

This field is available only for delimited standards. It identifies the maximum number of times composite or simple elements may repeat. Elements within composites, component elements do not repeat. The column displays automatically when a value greater than one appears for a composite or simple element. You may manually display the column, which you must do if you want to define repetitions for a delimited proprietary standard. To manually display the column, right-click in the detail area and select the **Show Repeats** command from the menu.

Type

This field is display only. You specify it in the Element window.

Min

This field is display only. You specify it in the Element window.

Max

This field is display only. You specify it in the Element window.

Procedures (Segment)

The following procedures are some basic tasks you may perform from the Segment window.

To Add a Segment

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 From the **File** menu, select **Add**.
The **New Segment** dialog box appears.
- 3 Enter the name of your new segment, and select **OK**.

The Segment window appears.

- 4 Enter a description for your segment, and add elements to the segment. For instructions, refer to the topic, *To Add an Element to a Segment* (on page 628).
- 5 After you have added elements to your segment, do one of the following:
 - Close the window to save the information
 - or –
 - To add another segment:
 - Press **F6**
 - or –
 - Select **Add** from the **File** menu.

The **New Segment** dialog box re-appears and you can repeat steps 3-5.

To Modify an Existing Segment

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the list to view the appropriate standard and version, and select **Segments**.
- 2 In the right pane, double-click the name of the segment you want to modify.

The Segment window appears.

- 3 Make changes and close the window. The Workbench automatically saves the information.

To Save a Segment

The Workbench automatically saves records when you exit the window. You can also make an interim save by selecting **Save** from the **File** menu.

To Delete a Segment

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 From among the list of segments in the right pane, select the name of the segment you want to delete.
- 3 From the **File** menu, select **Delete**.

– or –



From the toolbar, select the **Delete** button

A **Confirm** dialog box appears.

- 4 Select **OK**.

To Sequentially Display Definitions of Segments

Sometimes you want to review the list of segment definitions. This procedure will display all definitions from all standards. The standard and version will change when you display definitions from a different standard. From the Segment window, to page backward and forward within the definitions:

- 1 From the **View** menu, select **Next** or press **F7** to see the next segment definition.
- 2 From the **View** menu, select **Prior** or press **F8** to see the previous segment definition.

To Delete an Element From a Segment

Make sure the Data Explorer window is open.

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 In the right pane, select the name of the segment whose element you want to delete. The Segment window appears.
- 3 In the detail area of the Segment window, use the **Tab** key or the arrow keys to position the cursor to the row that you want to delete.
- 4 Right-click, and select **Delete Row**.
A **Confirm** dialog box appears.
- 5 Select **OK**.

To Add an Element to a Segment


- 1 If you are already in the Segment window, proceed to step 2.
– otherwise –
From Data Explorer, choose the segment to which you want to add elements.
- 2 Place your cursor in the segments detail area, which is the box below the header area.
- 3 Create a blank line.

IMPORTANT: You may insert a line anywhere in the segment. They are listed in order based on the unique sequence numbers you use. To insert elements, you must have a gap in your sequence numbers. If you have no gap in your sequence numbers, you must manually resequence other elements to create a gap.

- To add a line at the end, place your cursor in the last line and press the **Tab** key or the down-arrow key.
– or –
- To insert a line before another line, place your cursor on the detail line, and

- Press the **Insert** key
 - or –
 - Right-click, and select **Insert Row** from the menu
- 4 In the blank line, enter the appropriate information, which may include creating a new element.

NOTE: When you move your cursor to a different line, the Workbench will display the elements in the correct order.

- a) In the **Seq (on page 623)** column, type a sequence number.
 - b) To add a new element:
 1. In the **Ele ID** (on page 623) column, type the element ID, press **Tab**, and then double-click quickly.
 2. When the Element window appears, type the appropriate information for the element, and select **OK**.
 - c) To select an existing element:
 1. In the **Ele ID** (on page 623) column, right-click once.
 2. Click the selection button, , when it appears.
 3. From the **Select Element** dialog box, select the appropriate element and click **OK**.
 - d) In the **Rqmt** (on page 626) column, type or select the requirement designator.
- 5 Close the window to save your changes.

To Assign Maximum Repeats to a Composite or Simple Element in Delimited Standards

Within a delimited standard, the Workbench allows you to assign a maximum number of repetitions to a composite element or simple element within a segment. You do this from the Segment window. To do this, you must have identified the standard as a delimited standard on the Standard Version Properties window. You must also have identified a repetition separator on the **Service Characters** tab of the Wrapper Properties window.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the name of the segment to whose element you want to assign repetitions.

The Segment window appears.
- 3 In the detail area of the Segment window, right-click and select **Show Repeats**.

The **Rep** column appears.
- 4 Enter the maximum number of repetitions for the composite or simple element in the Rep column.
- 5 Close the window. The Workbench automatically saves the changes.

To Assign Segment Tags For Non-initial Elements In Fixed-Length Segments

The Workbench optionally allows you to assign something other than the first element of a fixed-length segment as the segment tag. You do this from the Segment window. To do this, you must first explicitly identify the segment as a fixed-length segment.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the name of the segment whose element you want to assign as the segment tag.
The Segment window appears.
- 3 From the **File** menu, select **Properties**.
- 4 From the **General** tab of the Segment Properties window, select the category of **Fixed**.
- 5 Select **OK** to close the Segment Properties window.
- 6 From the Segment window, use the **Tab** key or the arrow keys to position the cursor in the **Tag** column of the element that you want to assign as the tag.
- 7 Press any key and a **T** (tag) will appear in the column.
If you make a mistake, delete the **T** by selecting it and pressing any key. There can only be one element assigned as a tag, so if you try to assign another element as a tag also, the first element will lose its **T** and the new element will have one.
- 8 Close the window. The Workbench automatically saves your changes.

To Assign an Internal Field To an Element

You can tell the TRM to store certain information for future reference during mapping when it parses the incoming data, usually from a wrapper. You do this by assigning a predefined internal field location to an element in a Segment window.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the name of the segment to whose element you want to assign an internal field.
The Segment window appears.
- 3 In the detail area of the Segment window, right-click, and select the **Show Fields** command.
The **Field** column appears.
- 4 In the **Field** column of the target element, click twice slowly. (If you double-click quickly, you will go to the Element window.)
- 5 Click the down arrow to display the list of internal fields.
- 6 Scroll through the list and select the appropriate field for the element.

- 7 Close the window. The Workbench automatically saves the changes.

To Add Edibasic Validate Routines to Segments

You can add your own validation routines to segments. Although the TRM does some compliance checking, you may want to perform additional validations. The TRM will execute the routine during parsing whenever it encounters this document. The validation routine may be overridden by another validation routine written for this segment from the Document window. You should always review validation routines by printing a document report, so you will know which ones will take effect.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the desired segment.
- 3 Place your cursor in the Segment window header area (top part of the window).
- 4 Right-click your mouse button, and select the **Validate** option.
The Edibasic Edit window appears with **Method Validate** header and trailer code.
- 5 Enter your validation routine between the header and trailer code, which typically includes using the **Exception** function.
- 6 *Check your syntax for errors* (on page 585).
- 7 *Print a document report* (on page 571) to review the validation routines.

To Add Edibasic Validate Routines to Elements within a Segment

You can add your own validation routines to simple or composite elements within segments. Although the TRM does some compliance checking, you may want to perform additional validations. The TRM will execute the routine during parsing only when this segment occurs within this document. The validation routine may override another validation routine written for this element from the Element window or for a composite from the Composite window. You should always review validation routines by printing a document report, so you will know which ones will take effect.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the desired segment.
- 3 Place your cursor in the detail area and select the element to which you will attach the validation routine.
- 4 Right-click your mouse button, and select the **Validate** option.
The Edibasic Edit window appears with Method Validate header and trailer code.
- 5 Enter your validation routine between the header and trailer code, which typically includes using the **Exception** function.

- 6** *Check your syntax for errors* (on page 585). When you exit the window a check mark appears in the blank column to indicate that code exists for it.
- 7** *Print a document report* (on page 571) to review the validation routines that may be invoked for the document.

To Declare Variables for Segments

You can declare local variables or global variables to use within validation routines for segments. You can declare local variables on the Variables page or within the method on the Validate page. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the Variables page only. They have a lifetime from the point of declaration to the end of processing for the input stream. This allows you to store information based on validation routines during the parsing process and use it later to create a proprietary acknowledgment, for example.

Make sure the Data Explorer window is *open* (on page 532).

- 1** In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2** In the right pane, double-click the segment for which you want to create the variable.
- 3** Place your cursor in the window header area (top part of the window).
- 4** Right-click, and select **Variables**.

The Edit window appears with the Variables page selected.

- 5** Enter your local or global variable declaration statement here.

The following example declares a global variable:

```
Global Gmyvar as Integer
```

The following example declares a local variable:

```
Dim Lmyvar as Integer
```

- 6** *Check your syntax for errors* (on page 585).

To Declare Variables for Elements within a Segment

You may declare local variables or global variables to use within validation routines for elements within a specific segment. You declare local variables on the Variables page or within the method on the Validate page. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the Variables page only. They have a lifetime from the point of declaration to the end of processing of the input stream. For example, you may store information based on validation routines during the parsing process and use it later to create a proprietary acknowledgment.

Make sure the Data Explorer window is *open* (on page 532).

- 1** In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2** In the right pane, double-click the segment for which you want to create the variable.
- 3** Position your cursor in the element detail area and select the appropriate element.

- 4 Right-click, and select **Variables**.

The Edit window appears with the Variables page selected.

- 5 Enter your local or global variable declaration statement here.

The following example declares a global variable:

```
Global Gmyvar as Integer
```

The following example declares a local variable:

```
Dim Lmyvar as Integer
```

- 6 **Check your syntax for errors** (on page 585).

- 7 Close the window.

A check mark appears next to the element to indicate code exists for it.

Segment Properties Window

The Segment Properties window allows you to identify a category, if you need to change the processing for this segment from that defined at a higher level. It also allows you to enter conditions, and it displays which documents or wrappers use this segment. To delete a condition, select the condition and press **CTRL+DELETE**.

Procedures (Segment Properties)

The following procedures are some tasks you may perform from the Segment Properties window.

To Add a Condition to a Segment

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2 In the right pane, double-click the segment to which you want to add conditions.

The Segment window appears.



- 3 From the toolbar, click the **Properties** button,

The Segment Properties window appears.

- 4 Select the **Conditions** tab, place your cursor in the detail area, and proceed as follows:
 - a) In the **Seq** column, enter a sequence number, which typically is the same as the element within the segment to which the condition applies.
 - b) To enter multiple conditions for an element, in the **Sub Seq** column, enter a number that together with the **Seq** number uniquely identifies the condition.

- c) In the **Condition** column, click twice slowly to display an arrow, and click the arrow and choose one of the types of conditions from the list:

P	Paired Condition	If any data element specified is present, then all must be present.
R	Required Condition	At least one of the data elements specified must be present.
E	Exclusion Condition	Not more than one of the data elements specified may be present.
C	Conditional Condition	If the first data element specified is present, then all others must be present.
L	List Conditional Condition	If the first data element specified is present, then at least one of the others must be present.

- d) In the blank columns, enter the sequence numbers of the elements to which the conditions apply. They must be the same as the sequence numbers in the elements detail area in the Segment window.
- e) To add more conditions, tab to create a new row and repeat steps a-e.

- 5** Select **OK** to close the window to save the information.

To Delete a Condition From a Segment

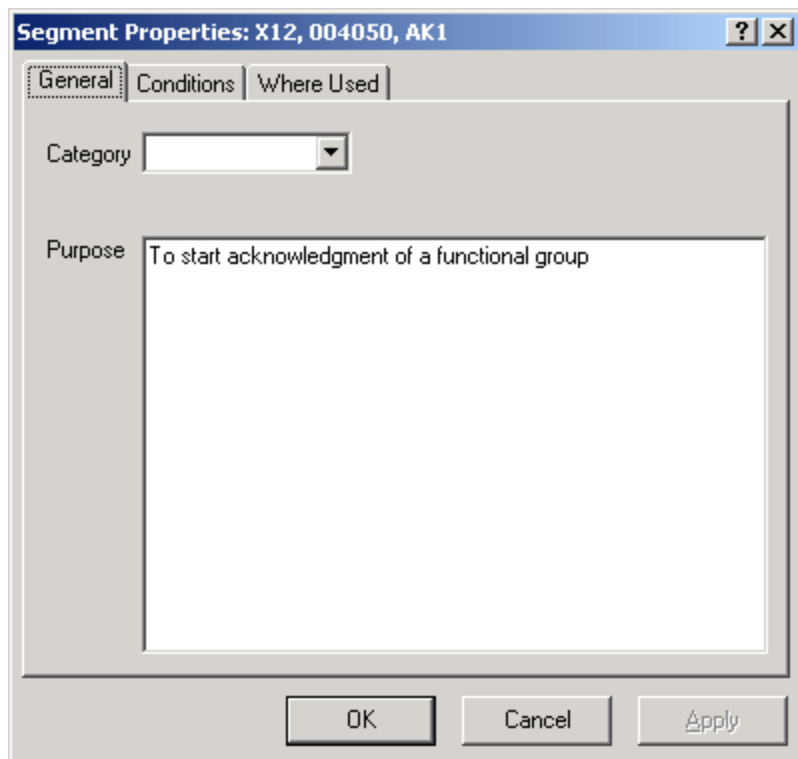
Make sure the Data Explorer window is *open* (on page 532).

- 1** In the left pane, select **Segments** from the expanded view of the appropriate standard and version.
- 2** In the right pane, double-click the name of the segment whose condition you want to delete.
The Segment window appears.
- 3** From the **File** menu, select **Properties**.
The Segment Properties window appears.
- 4** From the Segment Properties window, select the **Conditions** tab.
- 5** Select in the condition you want to delete.
- 6** Press **CTL+DELETE** from the keyboard.
A **Confirm** dialog box appears.
- 7** Select **OK**.

8 Select **Apply** or **OK** to save the change.

(Segment Properties) General

The **General** tab allows you to specify a parse and generate category for the segment, when you need to override a higher-level category, and to write any important information that you want to keep about the segment.



General Tab (Segment Properties Window)

Category

The category identifies the parsing and generation routines that will be used for the segment, if different from that of the document. If you identify a category at this level, it applies to all elements within this segment, if you do not override it with another category at a lower level. The options available are as follows:

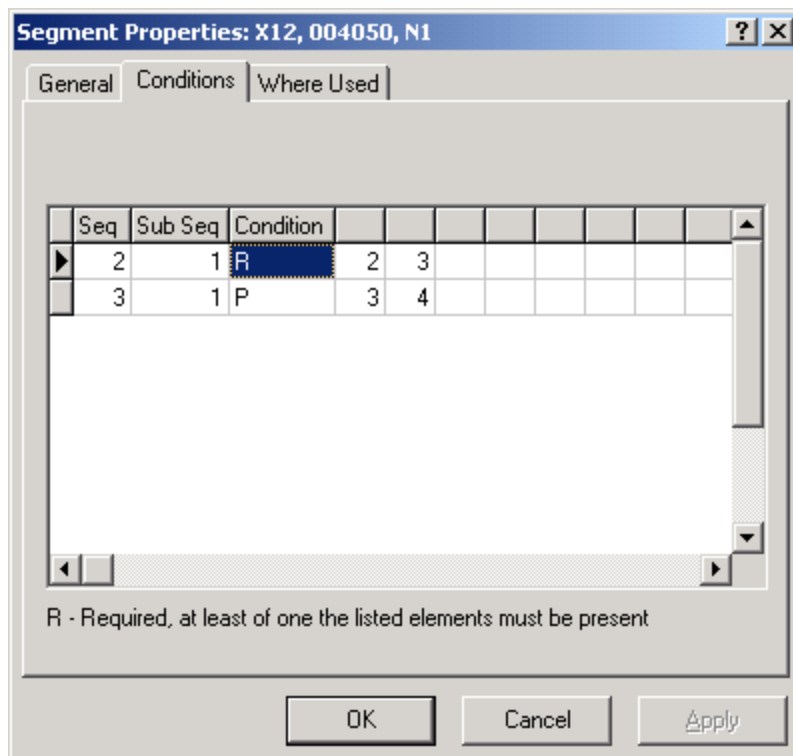
Type	Description
Delimited	Delimited routines use delimiters to parse the data, and generate all data entities separated with delimiters, unless otherwise noted. The delimiters used are the ones defined for the wrapper set that is associated with the interchange at the time of translation.
Fixed	Fixed routines parse the data based on the length of fields and generate the data to the full length of the field. The tag can be at any offset. The tag column only displays when the segment is identified as fixed.
SWIFT	Identifies segments within a SWIFT document. It consists of a segment tag within colons (:tag:) followed by zero or more SWIFT elements. During parsing, it ignores any carriage-return, linefeed (CR-LF) characters that occur at the end of the segment. During generation, if no elements have been generated for the segment, CR-LF is added to the end of the segment. This segment type is only valid for SWIFT documents. Do not use this type for SWIFT wrapper segments. Defining a SWIFT type for an element used in wrappers will produce undefined results.
SWIFT Block	<p>identifies segments within a SWIFT wrapper, which can be one of four types:</p> <p>text block header, consisting of {tag:CRLF with no trailing bracket</p> <p>text block trailer, consisting of -}</p> <p>fixed, if the first element in the segment is of the category SWIFT Fixed. The elements within the block occur in a prescribed order (positional), but trailing elements may be omitted or truncated with the block terminator (}).</p> <p>block, if the first element in the segment is of the category SWIFT Block. The elements may occur in any order and are identified by their preceding tags.</p> <p>This segment type is only valid for SWIFT wrappers. Do not use this type for SWIFT document segments. Defining a SWIFT type for a segment used in documents will produce undefined results.</p>
XML	Generates XML syntax. The description for the first segment in a loop becomes the tag for that group of elements. Subsequent segment descriptions become tags for the nested elements.

Purpose

The purpose field gives you up to 240 characters of space to describe the purpose of the segment or any other pertinent notes. This field is optional.

(Segment Properties) Conditions

The **Conditions** tab allows you to enter information about elements that are part of this segment. The TRM will compare the incoming data with these conditions for compliance.



Conditions Tab (Segment Properties Window)

Seq

This sequence number (**Seq**) together with the sub-sequence (**Sub Seq**) number uniquely identifies the condition. Typically, the sequence number is the same as the element within the segment to which the condition applies. This is not mandatory, but it makes troubleshooting easier.

Sub Seq

The sub-sequence (**Sub Seq**) number together with the sequence (**Seq**) number uniquely identifies the condition. The sub-sequence number allows you to enter multiple conditions for a single element.

Condition

The condition can be one of 5 types. Valid entries are as follows:

P	Paired Condition	If any data element specified is present, then all must be present.
R	Required Condition	At least one of the data elements specified must be present.
E	Exclusion Condition	Not more than one of the data elements specified may be present.
C	Conditional Condition	If the first data element specified is present, then all others must be present.
L	List Conditional Condition	If the first data element specified is present, then at least one of the others must be present.

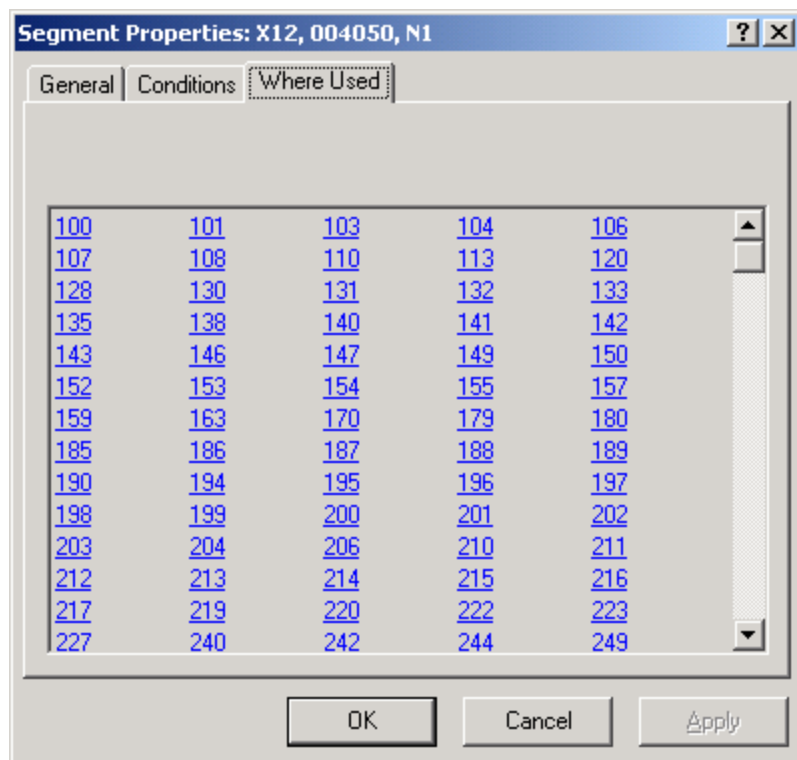
The blank columns are for the sequence numbers of the elements to which the conditions apply. They must be the same as the sequence numbers in the elements detail area in the Segment window.

Element Sequence Numbers

These are the sequence numbers of the elements to which the conditions apply. They must be the same as the sequence numbers in the elements detail area in the Segment window.

(Segment Properties) Where Used

The **Where Used** tab is automatically completed on your current configurations. It shows you all documents or wrappers that currently use this segment.



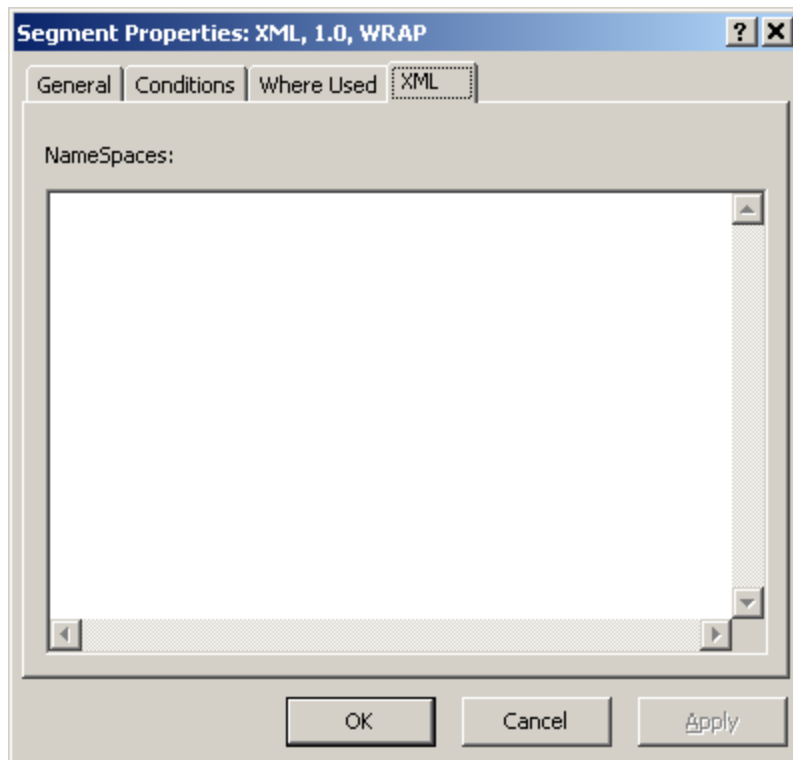
Where Used Tab (Segment Properties Window)

Where Used

The **Where Used** field shows you all documents or wrappers for which this segment is currently a valid segment. This information is maintained by the system and displays automatically based on your configurations. To hot-key to the definition of a particular segment displayed in the **Where Used** box, simply double-click over its name.

(Segment Properties) XML

The **XML** tab appears when **XML** is selected from the **Category** list on the **General** tab. This tab allows you to specify a namespace used for input or output.



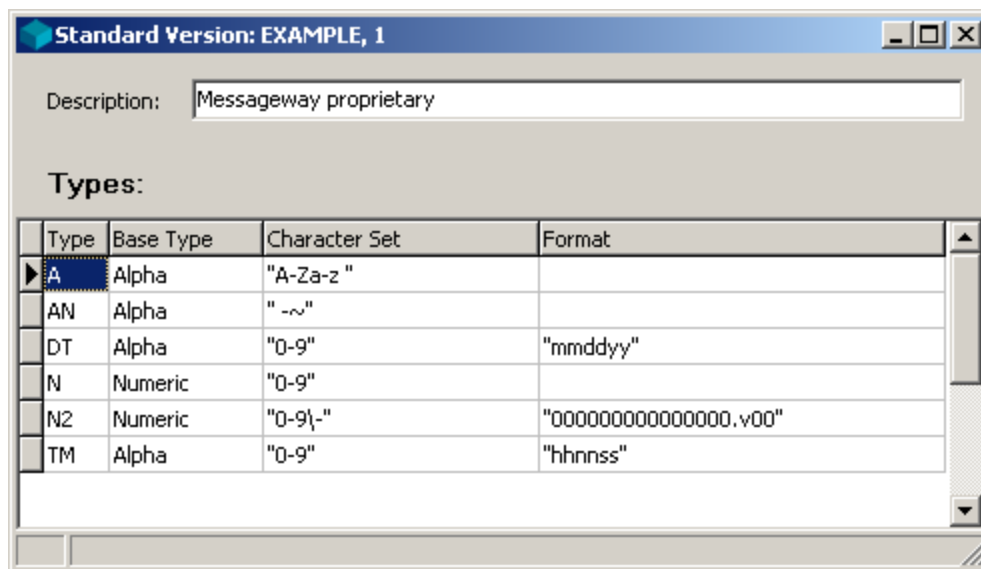
XML Tab (Segment Properties Window)

Namespace

Typically, you specify namespaces definition on the wrapper segment of the input wrapper that is generated by the `trmxml` user exit. The syntax of the command should be, `xmlns="ValidURI"`. For example, it might be, `xmlns="http://www.w3.org/TR/xml-names"`. Of course, this should be the location for your own particular namespace.

Standard Version Window

To specify data types that are valid for a given standard, you must use the Standard Version window. You can identify one set of data types for each standard and version, which becomes the profile name. You access this window in the right pane of Data Explorer by selecting a version in the left pane. You can also hotkey to this window from the Element window.



Standard Version Window

The Standard Version window has an associated *Standard Version Properties window* (on page 649). You can access the Standard Version Properties window from the Standard Version window by selecting **Properties** from the **File** menu.

Standard Name

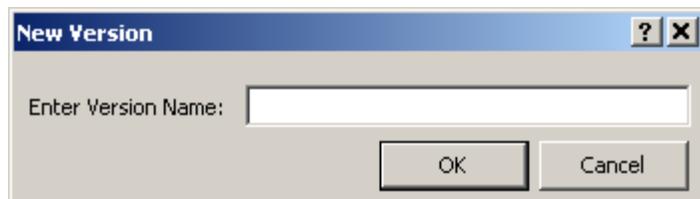
The Standard Name together with the Version Name uniquely identifies a standard version. You enter the name of the standard by selecting **Standards** from the left pane of Data Explorer and then **File>Add**. Enter a name of 1 to 8 alphanumeric characters when the **New Standard** dialog box appears.



New Standard Dialog Box

Enter Version Name

The version name together with the standard name uniquely identifies a standard version. Enter a name of 1 to 8 alphanumeric characters when the **New Version** dialog box appears.



New Version Dialog Box

Description

The description field allows you to enter a textual description up to 80 characters long for the standard and version.

Type (Types)

This field together with the standard and version uniquely identify the data type. It can contain 1 to 2 alphanumeric characters. This is the value you use to identify the data type of an element. It is the base type, however, that controls how data is treated during parse and generate.

Base Type (Types)

The base type of an element controls formatting during parse and generate. All element base types are Alpha, Numeric, ID, Binary, or DateTime. For internal processing, however, all elements are treated as strings. When required, you can convert data from one type to another using Edibasic functions and variables.

Type	Formatting Characteristics
Alpha	<ul style="list-style-type: none"> ▪ may contain any character included in a defined character set. ▪ parsing and validation: spaces are removed from the right of fixed-length fields. ▪ generation: spaces are added to the right of fixed-length fields. ▪ formats: if defined, are ignored
Numeric	<ul style="list-style-type: none"> ▪ may contain any character included in a defined character set. ▪ parsing and validation: leading plus signs and then spaces and zeros are removed from the left of fixed-length fields, and then spaces are removed from the right of fixed-length fields. ▪ generation: zeros are added to the left of fixed-length fields and then minus signs, as required. ▪ formats: if defined for both elements will be converted automatically.
ID	<ul style="list-style-type: none"> ▪ may contain any character included in a defined character set. ▪ parsing and validation: spaces are removed from the right of fixed-length fields. ▪ generation: spaces are added to the right of fixed-length fields.
Binary	<ul style="list-style-type: none"> ▪ may represent any character included in a defined character set, which would typically contain all 256 ANSI characters, values 0 through 255. Actual values should be binary 00000000 through 11111111. ▪ parsing and validation: no compression. ▪ generation: no padding.
DateTime	<ul style="list-style-type: none"> ▪ may contain any character included in a defined character set. ▪ parsing and validation: spaces are removed from the right of fixed-length fields. ▪ generation: spaces are added to the right of fixed-length fields. ▪ formats: if defined for both elements will be converted automatically.

Character Set (Types)

This identifies the characters that are valid for this type. The TRM performs validation during parsing to ensure that only these characters are used for the type. It generates a compliance error if an element contains any characters not listed as valid characters for its type. You can enter from 1 to 128 characters in this field. Usually, you enter a range of characters in the form of the two ends of the sequence separated by a dash, (-). If you specify more characters than those within one range of values, you must enclose the string in single or double quotation marks. To identify the dash (-) as a character and not meaning "through", it must be the last character of the string. For non-printing characters, you must enter a hexadecimal value in the form, `\xnn`, where `nn` is the hex value of the character. Here are a few examples:

CharSet Entries	Description
"\x00-\xff"	all 256 ASCII values, identified by the hex values of the first and last characters.
" ~"	all characters from space through the tilde (~). The first character after the double-quote is a space.
"0-9.+-"	all numeric characters and the period, plus, and minus signs.
0-9	all numeric characters (quotes not required for a single range of characters).
E" ~"	all EBCDIC characters that translate to any printable ASCII characters from space through the tilde (~). This is not the same as all printable EBCDIC characters.
E"HDR"	EBCDIC characters matching ASCII values, such as "HDR". Use this type of character set to match incoming EBCDIC literal values, for example, a header segment tag. Make sure there are no spaces in the incoming segment tag. The ASCII values for tags, such as "HDR", as specified in a standards definition, will be converted to EBCDIC before a match is attempted to identify the incoming standard. This character set will also generate EBCDIC output.
BCD	all data is packed Binary Coded Decimal (BCD). ASCII values will be converted to or from BCD values. The format statement identifies whether you have signed or unsigned data or implicit decimals.

NOTE: The tables to convert EBCDIC to and from ASCII use IBM code page 00038 (US-ASCII).

Format (Types)

This field can contain from 1 to 32 alphanumeric characters. If a DateTime or Numeric data type has a format statement associated with it, the TRM will potentially be able to automatically reformat the data. For automatic reformatting to occur, both input and output elements must have DateTime or Numeric data types with associated formats. You can also control formats within Edibasic using various functions in your mapping methods, such as, **Format\$**. Refer to the description of *this function* (on page 737) for a detailed explanation of your options.

NOTES: If you enter these format statements with uppercase characters, except **AM** and **PM**, the TRM interprets them as constants, and those values become part of the generated data.

If the automatic formatting between DateTime or Numeric data types does not produce the desired results, use the **Format\$** function in the MapEle statement, which overrides the automatic formatting.

Procedures (Standard Version)

These procedures describe some tasks you can perform from the Standard Version window.

To Add a Standard

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Standards**.
- 2 From the **File** menu, select **Add**.
The **New Standard** dialog box appears.
- 3 Type the name of your new standard.
- 4 Select **OK**.

You return to Data Explorer, where you must now enter the name of a version for the standard. Refer to the procedure, *To Add a Standard Version* (on page 645).

To Add a Standard Version

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select the standard from the expanded view of the Standards folder.
- 2 From the **File** menu, select **Add**.
The **New Version** dialog box appears.
- 3 Enter the name of your new version.
- 4 Select **OK**.

You return to Data Explorer, where you must now specify properties of the version for the standard. Refer to the various procedures associated with *Standard Version Properties window* (on page 649), under the topic *Procedures* (on page 651).

To Define Standard Data Types for a Standard Version

Data types control compliance checking of the input data and formatting of the input and output data for a particular standard version.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Standard Version Profile** from the expanded view of the appropriate standard version.
- 2 In the right pane, double-click the version.
The Standard Version window appears.
- 3 Enter the types you need for this standard version to control compliance checking and formatting.
 - a) In the *Type* (on page 642) column, type one or two characters to represent the definition.
 - b) In the *Base Type* (on page 643) column, choose the type from the list, which will control how this data is handled internally.
 - c) In the *Character Set* (on page 644) column, type the set of valid characters for this data type, which will be used to validate the incoming data.
 - d) Optionally, in the *Format* (on page 645) column, type the format, based on options listed for the *Format\$* (on page 737) function.
- 4 Close the window. The Workbench automatically saves the information.

To Define EBCDIC Data Types for a Standard Version

Data types control compliance checking of the input data and formatting of the input and output data for a particular standard version. EBCDIC is supported for fixed-format standard versions. An alternative to this method of handling EBCDIC data would be to use XML.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Standard Version Profile** from the expanded view of the appropriate standard version.
- 2 In the right pane, double-click the version.
The Standard Version window appears.
- 3 Enter the types you need for this standard version to control compliance checking and formatting.
 - a) In the *Type* (on page 642) column, type one or two characters to represent the definition.
 - b) In the *Base Type* (on page 643) column, choose the type from the list, which will control how this data is handled internally.

- c) In the *Character Set* (on page 644) column, type the character **E** followed by the set of valid characters for this data type within double quotation marks.
For example, **E"0-9"**, will cause the EBCDIC values zero through nine to be converted to or from ASCII.
 - d) Optionally, in the *Format* (on page 645) column, type the format, based on options listed for the *Format\$* (on page 737) function.
- 4 Close the window. The Workbench automatically saves the information.

To Define BCD Data for a Standard Version

Data types control compliance checking of the input data and formatting of the input and output data for a particular standard version. Binary Coded Data (BCD) packs numeric data, two numbers per byte or a number and a sign. The length of the field or element is based on the bytes, not the number of characters in those bytes.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, select **Standard Version Profile** from the expanded view of the appropriate standard version.
- 2 In the right pane, double-click the version.
The Standard Version window appears.
- 3 Enter the types you need for this standard version to control compliance checking and formatting.
 - a) In the *Type* (on page 642) column, type one or two characters to represent the definition.
 - b) In the *Base Type* (on page 643) column, select **Numeric** from the list, since BCD is packed numeric data.
 - c) In the *Character Set* (on page 644) column, type the characters **BCD** without double quotation marks. This cause the BCD values to be converted to or from ASCII.
 - d) In the *Format* (on page 645) column, based on options listed for the *Format\$* (on page 737) function, specify whether the data is unsigned (***0**) or signed (***0+-**) or whether it uses implied decimals, for example for monetary values (***0.v00**).
- 4 Close the window. The Workbench automatically saves the information.

To Modify Standard Data Types

Make sure the Data Explorer window is open.

- 1 In the left pane, expand the folders to view of the appropriate standard version, and select **Standard Version Profile**.
- 2 In the right pane, double-click the standard version.
- 3 Make changes.
- 4 Close the window. The Workbench automatically saves the information.

To Save a Standard Version Profile

The Workbench automatically saves records when you exit the window. You can also make an interim save by selecting **Save** from the **File** menu.

To Delete a Standard Version

You cannot delete a folder for an entire standard. You must first delete each standard version. When you delete the last standard version, the Workbench also removes the folder for the standard. When you delete a standard version, you are actually deleting the standard version and any documents, segments, elements, and composite elements defined for that version.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, choose the version from within the expanded view of the appropriate standard.
- 2 From the **File** menu, select **Delete**.

– or –



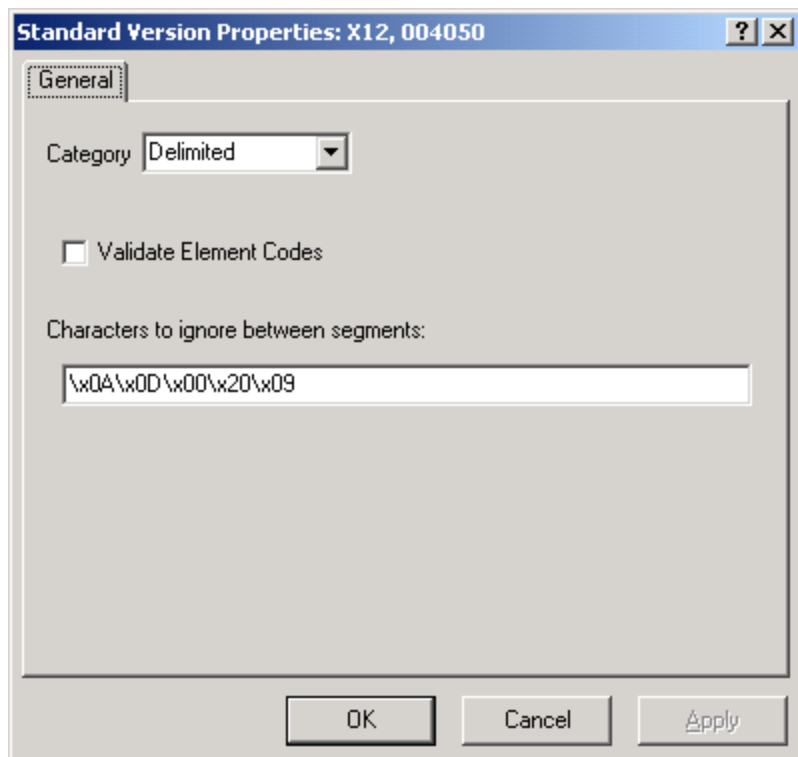
From the toolbar, select the **Delete** button

A **Confirm** dialog box appears.

- 3 Select **OK**.

Standard Version Properties Window

The Standard Version Properties window allows you to specify characteristics for the entire standard: a category for parsing and generation strategies, whether you want to validate element codes, and characters that should be ignored during parsing.



Standard Version Properties Window

Category

The category identifies the parsing and generation routines that will be used for the standard. You can specify routines from this point down in the structure. If you identify a schema at this level, it will be applied to all lower entities within this standard, if you do not override it with another schema at a lower level. EBCDIC is only supported for fixed-length data. The options available are as follows:

Type	Description
Delimited	Delimited routines use delimiters to parse the data, and generate all data entities separated with delimiters, unless otherwise noted. The delimiters used are the ones defined for the wrapper set that is associated with the interchange at the time of translation.
Fixed	Fixed routines parse the data based on the length of fields. They generate the data by padding to the full length of the field.
SWIFT	This selection invokes SWIFT parsing and generation routines for all lower categories, unless explicitly overridden at a lower level.
XML	This selection invokes XML parsing and generation routines for all lower categories.

Validate Element Codes

Selecting this check box will allow the TRM to check values of element codes on any elements in the standard for which codes are defined. If the match fails, the TRM will reject the document or wrapper. You can check codes for a specific document or wrapper by selecting a similar option in the Document Properties window or Wrapper Properties window. When you clear or check this box, the corresponding boxes on all documents and wrappers are cleared or checked, respectively.

This box can have three states as follows:

State	Description
<input type="checkbox"/> Validate Element Codes	No check mark means that the TRM will not validate any element codes for the standard version.
<input checked="" type="checkbox"/> Validate Element Codes	Dark check mark means that the TRM will validate element codes for all wrappers and documents for the standard version.
<input checked="" type="checkbox"/> Validate Element Codes	Dimmed check mark and box means that at least one, but not all, of the wrapper or document definitions defined for this version has the Validate Element Codes box checked on its Properties window.

Characters to Ignore Between Segments

The TRM will use the characters, specified as hexadecimal values in this field, when it parses the input message to ignore certain types of information sent between segments that should not be part of the data. These values are ignored when the IO Mode is set to **Binary** on the Wrapper Properties window.

The typical set of characters are `\x20\x09\x0A\x0D\x00`, which ignore white space, tab character, newline character, carriage return, and binary zero (null), respectively. For EBCDIC data you should use the EBCDIC hexadecimal values, `\x40\x05\x15\x25\x0D\x00`. You can change these values when necessary.

If you wish to follow the X12 or EDIFACT standards strictly, then you should clear this field for those standards. These standards do not allow white space between segments. However, we have provided this capability to avoid regression problems and to allow for those who do not strictly adhere to the standard.

Procedures

You can perform the following tasks from the Standard Version Properties window.

To Specify Parsing and Generation Category for a Standard Version

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the appropriate standard and version, and then choose **Standard Version Profile**.
- 2 In the right pane, double-click the version.



- 3 From the toolbar, select the **Properties** button.
- 4 From the **Categories** list, choose the routine that will correctly parse and generate documents within this standard version: **Fixed**, **Delimited**, **SWIFT**.
- 5 Select **OK**.

To Check Element ID Codes for an Entire Standard Version

You can invoke element code validation at the standard, document, or wrapper levels. If codes are already being checked at the standard version level, the status of this document or wrapper check box has no effect.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the appropriate standard and version, and then select **Standard Version Profile**.
- 2 In the right pane, double-click the version.



3 From the toolbar, select the **Properties** button.

4 Select the **Validate Element Codes** box.

A check mark appears.

Select **OK**.

To Ignore Specific Characters between Segments During Parsing

This procedure allows you to specify certain characters for the TRM to ignore that may appear between segments in incoming data, and not treat these characters as part of the data. This is particularly important when you use two characters to terminate your segments with *Text mode* (on page 681).

Make sure the Data Explorer window is *open* (on page 532).

1 In the left pane, expand the folders to view the appropriate standard and version, and then select **Standard Version Profile**.

2 In the right pane, double-click the version.



3 From the toolbar, select the **Properties** button.

4 Select the **General** tab, and in the box labeled **Characters to ignore between segments**, enter the hexadecimal value or values of those characters that you want to ignore. A typical sequence would be:

```
\x20\x09\x0A\x0D\x00
```

Each hexadecimal value must be preceded by **\x**.

5 Select **OK**.

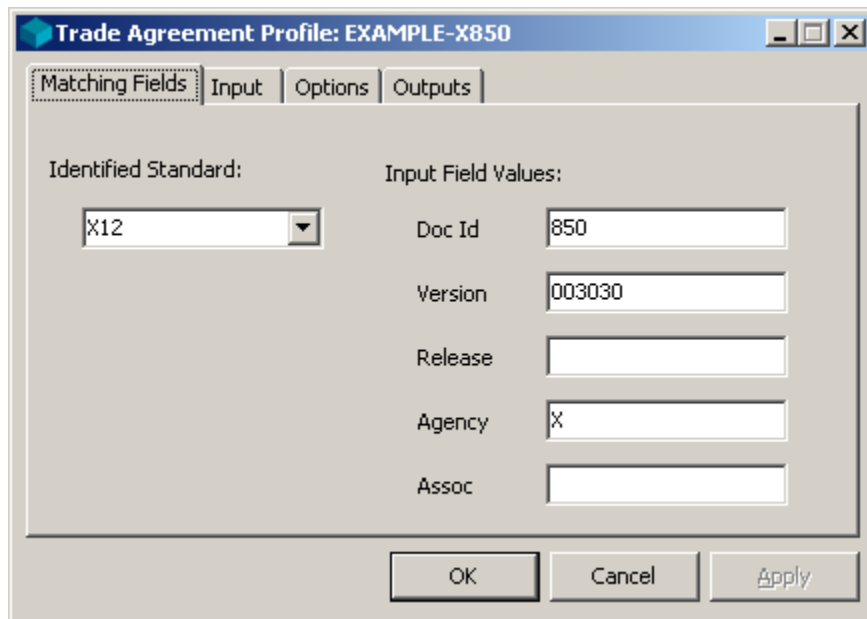
Trade Agreement Profile Window

The TRM needs this information to determine the various parameters for processing. You also use the Trade Agreement Profile to identify a security user exit that would be called when parsing an incoming document that requires security processing. You also use it to identify the user exit that would be called when generating an outgoing document that requires security processing. To support EDIFACT AUTACK, for example, you could use a security user exit to calculate a hash value to compare with the incoming value, or to generate as an outgoing value. You access this window from Data Explorer by selecting **Trade Agreements** in the left pane.

The **Outputs** page of the Trade Agreement Profile window has an associated *Trade Agreement Output Properties Window* (on page 660). From the **Outputs** tab, select the **Properties** button.

(Trade Agreement Profile) Matching Fields

The **Matching Fields** tab allows you to specify the matching criteria that the TRM will use to select a trade agreement.



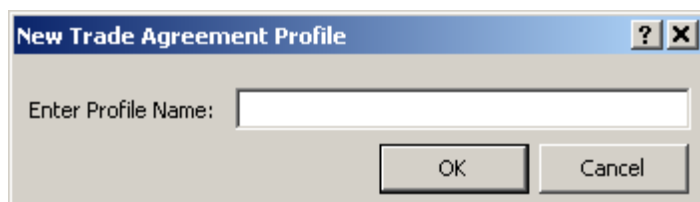
The screenshot shows a dialog box titled "Trade Agreement Profile: EXAMPLE-X850". It has four tabs: "Matching Fields" (selected), "Input", "Options", and "Outputs". Under "Identified Standard:", there is a dropdown menu showing "X12". Under "Input Field Values:", there are five text input fields: "Doc Id" with "850", "Version" with "003030", "Release" (empty), "Agency" with "X", and "Assoc" (empty). At the bottom are "OK", "Cancel", and "Apply" buttons.

Matching Fields Tab (Trade Agreement Profile Window)

Trade Agreement Profile Name

The name uniquely identifies the trade agreement profile. It can contain 1 to 32 alphanumeric characters. It can have one or more output trade agreements defined for it, each of which can have multiple translation outputs based on different maps.

To enter the profile name, select Trade Agreements from the left pane of Data Explorer, and then from the **File** menu, select **Add**. The **New Trade Agreement Profile** dialog box appears.



The screenshot shows a dialog box titled "New Trade Agreement Profile". It has a question mark icon and a close icon in the title bar. It contains a text input field with the label "Enter Profile Name:". At the bottom are "OK" and "Cancel" buttons.

New Trade Agreement Profile Dialog Box

Identified Standard

This is the standard of the incoming wrapper. Type the name of a standard or select one from the list. The value here must match the standard of the wrapper that was identified during the standards identification process.

Doc Id (Input Field Values)

This is the document ID of the incoming document. The value may be up to 8 characters long. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Document Id**. You assign the field in the element detail of the Segment window. If no element is assigned to **Document Id**, then the internal field will be null and will not match the value in this **Doc Id** field. For EBCDIC data, only **Doc Id** is explicitly supported. For XML data, **Doc Id** is ignored.

Version (Input Field Values)

This is the interchange version of the standard identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Document Version**. You assign this field in the element detail of the Segment window. If no element is assigned to **Document Version**, then this internal field will be null, and will not match the value in this **Version** field. For EBCDIC data, only **Doc Id** is explicitly supported.

Release (Input Field Values)

This is the release of the standard identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Release**. You assign this field in the element detail of the Segment window. If no element is assigned to **Release**, then this internal field will be null, and will not match the value in this **Release** field. For EBCDIC data, only **Doc Id** is explicitly supported.

Agency (Input Field Values)

This is the agency identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Agency**. You assign this field in the element detail of the Segment window. If no element is assigned to **Agency**, then this internal field will be null, and will not match the value in this **Agency** field. For EBCDIC data, only **Doc Id** is explicitly supported.

Assoc (Input Field Values)

This is the association identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Association Code**. You assign this field in the element detail of the Segment window. If no element is assigned to **Association Code**, then this internal field will be null, and will not match the value in this **Assoc** field. For EBCDIC data, only **Doc Id** is explicitly supported.

(Trade Agreement Profile) Input

The **Input** tab specifies the definitions used to parse the incoming document. It also specifies any security documents that should be present and any security user exits that should be performed.

The screenshot shows a dialog box titled "Trade Agreement Profile: EXAMPLE-X850" with four tabs: "Matching Fields", "Input", "Options", and "Outputs". The "Input" tab is selected. On the left, under "Inbound Document", there is a document icon and the following text: "Standard: X12", "Version: EXAMPLE", and "Doc Id: 850". On the right, under "Security Document:", there is an empty text input field. Below that, under "Security User Exit:", there is another empty text input field. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Input Tab (Trade Agreements Profile Window)

Inbound Document

This is the document definition used to parse the incoming document for document validation or translation. For an inbound XML document, the description of this document must match the tag of the root element. If you are only doing wrapper validation, you do not need to specify a document here.

Security Document

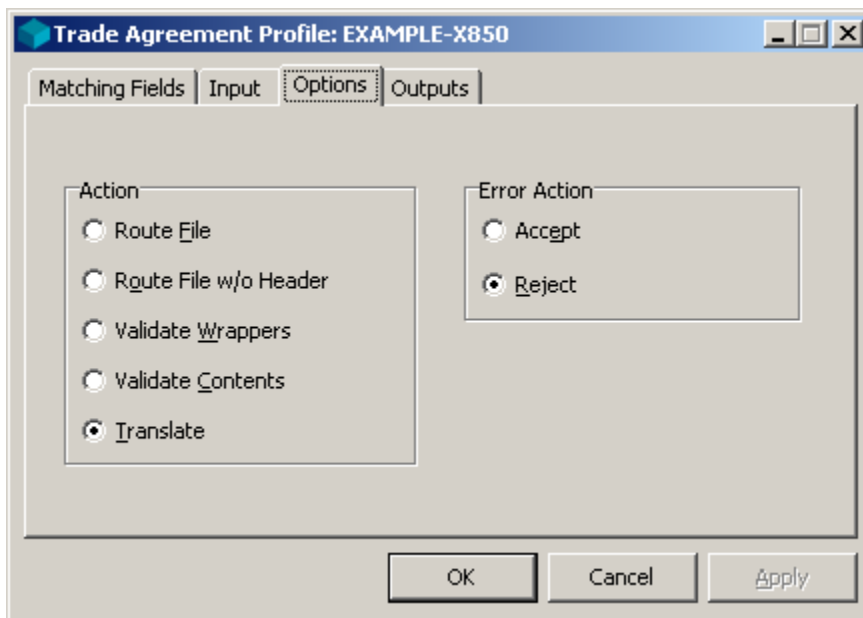
This field identifies the security document that is supposed to accompany the incoming document. If the document identified in this field does not accompany the source document in the same interchange, the TRM generates a security error. Leave this field blank if no security document is required.

Security User Exit

When the TRM receives documents, it will know to call a security user exit when there is a value in the **Security User Exit** field. This field identifies the name of the user exit that will be called to perform the security procedure. It must match the registered name you specify as the name parameter of the **TRMRegisterSecExit** function of your user exit. For additional information, refer to the *MW Translator User Exits Programming Manual*.

(Trade Agreement Profile) Options

The **Options** tab allows you to specify the work for the TRM to do.



Options Tab (Trade Agreement Profile Window)

Action

The action you choose identifies what type of processing you wish to perform. The options are as follows:

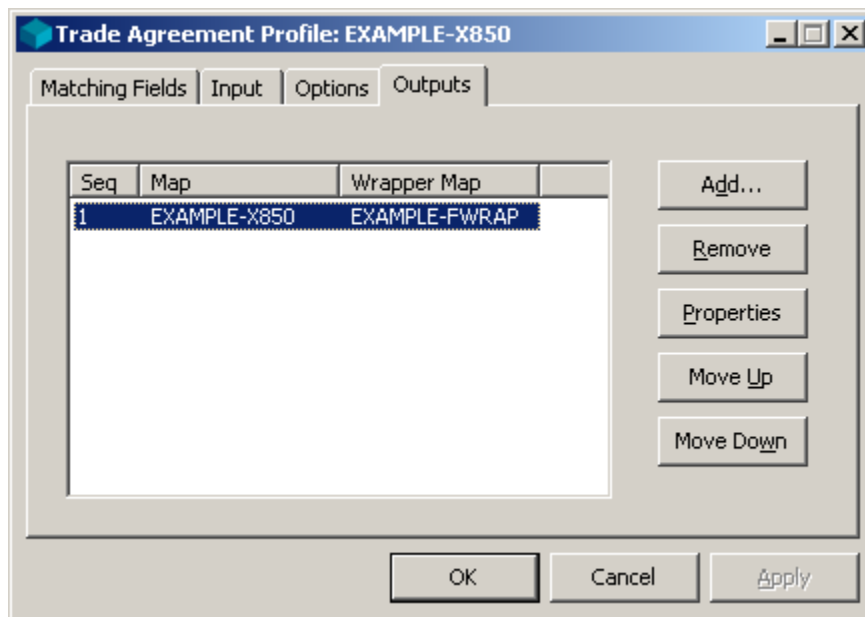
Action	Description
Route File	<p>Route a file whose content structure may not be defined, such as binary data. Use this option when you have no defined wrapper but want to route data using standard identification, typically with a special input location. The trade agreement profile should be associated with the standard ID record. Although the input file may or may not have a defined header, validate the header when one exists. Do not use this option to route standards whose wrappers contain more than a single header.</p> <p>This option is invalid for XML input.</p>
Route File w/o Header	<p>Route a file whose content structure may not be defined, such as binary data. The input file must have a header, which will be validated, and then removed before routing. For example, when the upstream process knows how to route data, it may put the routing information into any header format that the Translator Runtime Module (TRM) understands. In this case the TRM uses the header for routing and sends the rest of the file (minus the header) forward. This is important if the downstream process does not recognize headers.</p> <p>As above, the actual data that is routed can be any arbitrary binary data. For instance, when an upstream process wants to route an AutoCAD drawing to a downstream process that is expecting only the AutoCAD file, it can prepend a fixed header to tell the TRM where to route and send on to the TRM. The TRM will then remove the header and route the intact AutoCAD drawing to the destination. This will be especially useful with (but is not limited to) MessageWay for Windows using FTP.</p> <p>This option is invalid for XML input.</p>
Validate Wrappers	<p>Route a file whose wrapper structure is defined. Check wrapper data for compliance, and then route the data without modification.</p> <p>This option is invalid for XML input.</p>
Validate Contents	<p>Route a file whose wrapper and content structure are defined. Check wrapper and contents for compliance, and then route the data without modification.</p> <p>This option is invalid for XML input.</p>
Translate	<p>Translation implies validation of the wrappers and contents. In addition, this action performs translation(s) of the document based on the information in the Trade Agreement Output Properties associated with this trade agreement.</p>

Error Action

If there are errors during processing, this selection specifies whether to accept or reject the document. This information is written to the translation report and becomes part of a backward acknowledgment if one is configured for the sending partner or standard ID.

(Trade Agreement Profile) Outputs

The TRM allows you to process one input file multiple times using different translation profiles. The **Outputs** tab specifies the different profiles that will be used to translate the input document.



Outputs Tab (Trade Agreement Profile Window)

Outputs List box

This box lists all active translation definitions for this trade agreement profile. Each document to be translated has at least one entry in the list box that specifies what to do. You can create more than one translation profile for the incoming document. For example, for an incoming application purchase order, you might want to create one X12 purchase order to go to one trading partner as well as an EDIFACT purchase order to go to another trading partner. To do this, you create two translation profiles identifying each of the two output documents that belong to the one trade agreement representing the input document.

Each profile you identify is assigned a sequential number, so that the TRM knows how many translations it must perform and in what order for the given input file.

Add Button

The **Add** button allows you to add a translation profile to the list. The Trade Agreement Output Properties window appears.

Remove Button

The **Remove** button allows you to remove a translation profile from the list.

Properties Button

The **Properties** button allows you to change values in the Trade Agreement Output Properties.

Move Up Button

You can change the order of the translations by selecting the **Move Up** button, which simply moves the identified translation up in the sequential order. This allows you to group like documents for output in the same file.

Move Down Button

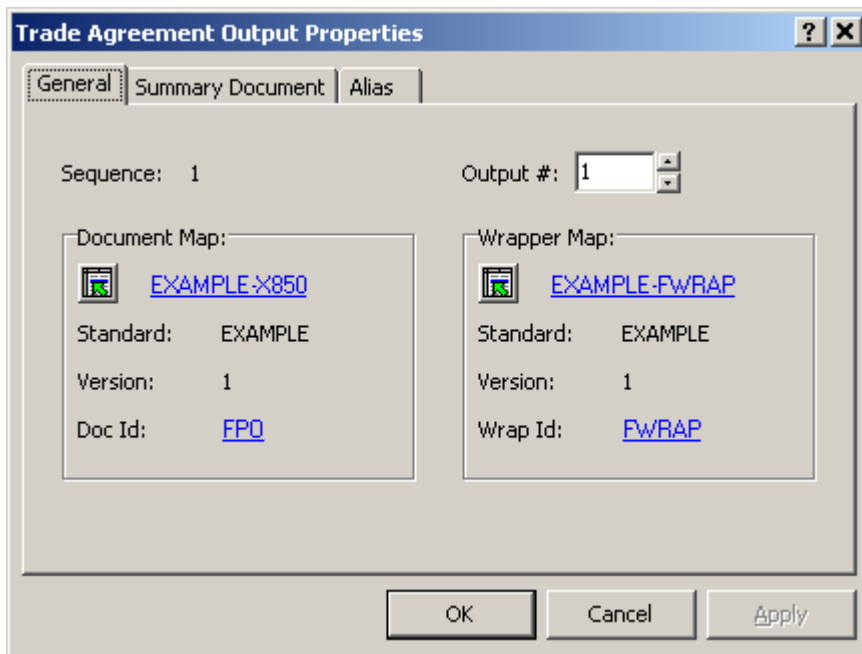
You can change the order of the translations by selecting the **Move Down** button, which simply moves the identified translation down in the sequential order. This allows you to group like documents for output in the same file.

Trade Agreement Output Properties Window

You access the Trade Agreement Output Properties Window from the **Outputs** tab of the Trade Agreement window. This is where you identify the document map, wrapper map, summary document information, and alias, if any, for each output of a translation. You can also specify that a particular output be placed in a file that is different from other types of output processed within an interchange.

(Trade Agreement Output Properties) General

The **General** tab allows you to specify the document map, wrapper map, and an output sequence number for specifying a stream for output documents.



General Tab (Trade Agreement Output Properties Window)

Sequence

Each profile you identify on the **Outputs** tab of the Trade Agreement window is assigned a sequential number, so that the TRM knows how many translations it must perform and in what order for the given input file. You can change the order of the translations only from the **Outputs** tab of the Trade Agreement window by selecting the **Move Up** or **Move Down** buttons, which simply moves the identified translation up or down in the sequential order.

Output

The output number identifies whether or not a particular document from an input file might be included with others in the same physical output file. Assuming other breaking rules do not take precedence, you can force a translated document to be placed in a different physical file from other types of output within the unit of work (interchange). You do so by giving it an output number that is different from the other outputs defined for any trade agreement used to process a particular interchange. This holds true across multiple trade agreement outputs. You may assign output numbers from **1** to **10**.

Document Map (Trade Agreement Output Properties)

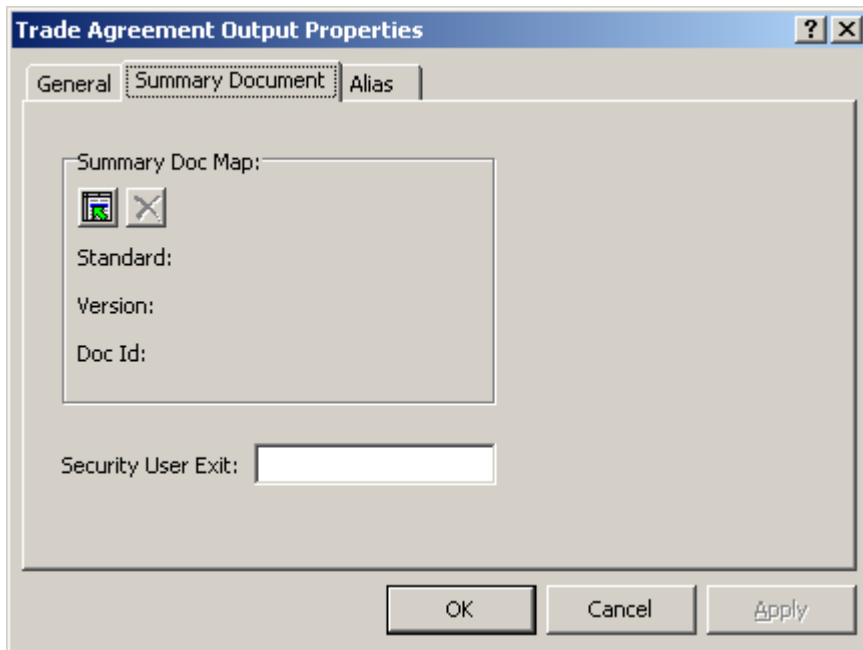
This identifies the name of the map used to translate the incoming document and the standard, version, and set definition of the output document.

Wrapper Map (Trade Agreement Output Properties)

This identifies the name of the map used to create the outgoing wrapper and the standard, version, and set definition of the output wrapper.

(Trade Agreement Output Properties) Summary Document

The **Summary Document** tab allows you to specify if you want a summary document to be sent with the outgoing interchange. A summary document responds to information in an interchange and is included in that interchange. Another use for a summary document is a forward acknowledgment.



Summary Document Tab (Trade Agreement Output Properties Window)

Summary Doc Map

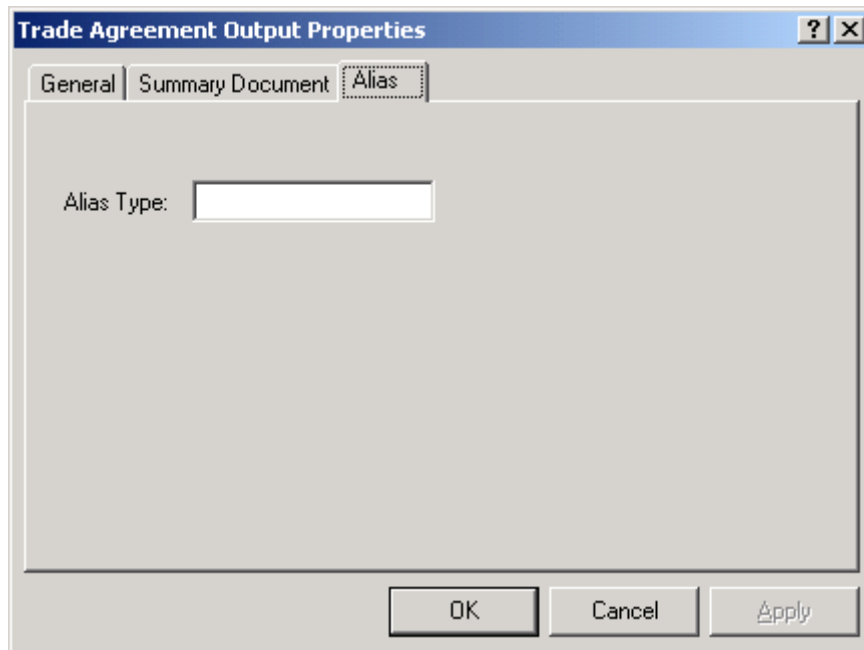
If you want to create a summary document for the interchange, you enter the map name here by using the browse button. The name of the map appears on the top, and the standard, version and document ID of the generated summary document appears below.

Security User Exit

When the TRM generates a document, it knows it must generate a summary document if a summary document map has been identified. If the TRM must also perform security procedures for this summary document, you enter the user exit in the **Security User Exit** field. This field identifies the name of the user exit that will be called to perform the security procedure. It must match the registered name you specify as the name option of the **ERMRegisterSecExit** function of your user exit. For additional information, refer to the *MW Translator User Exits Programming Manual*.

(Trade Agreement Output Properties) Alias

The **Alias** tab allows you to specify an alias type for this trade agreement output.



Alias Tab (Trade Agreement Output Properties Window)

Alias Type

This 8-character field allows you to identify alternative sender and/or recipient profiles for the outbound document. This provides alternative IDs for the outbound wrapper and an alternative location. This alias does not apply to backward acknowledgments, only documents going forward to the recipient of the original document.

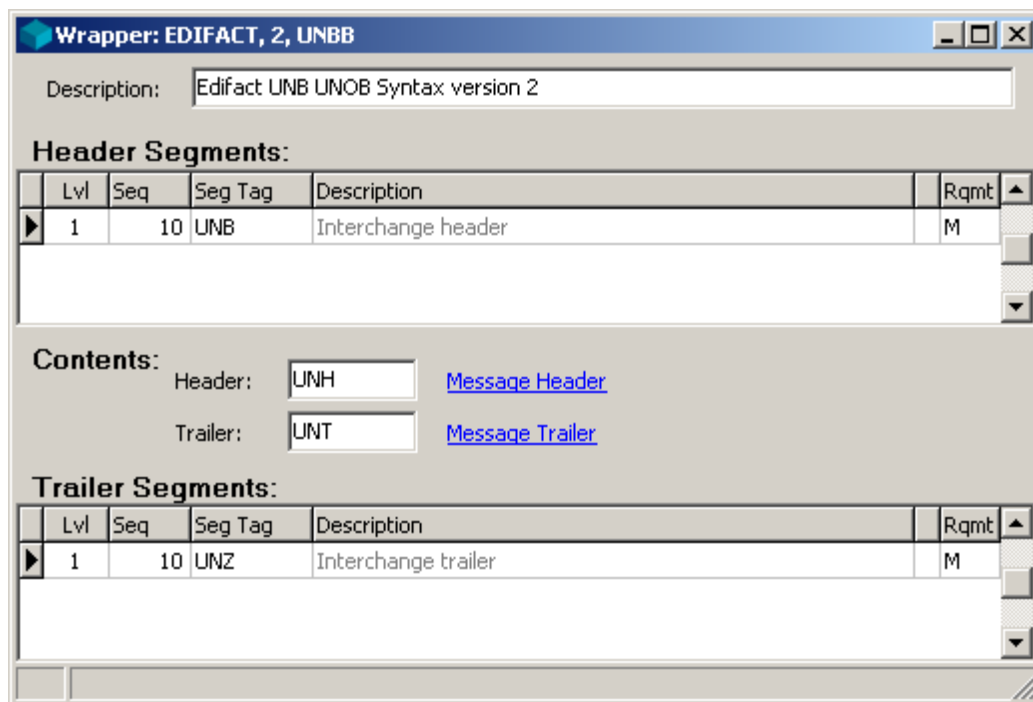
Using aliases requires that you define an additional partner profile as that alias type, and link it to the appropriate original profile by adding the aliased partner to the alias list. For more information, refer to the topic in the chapter "Defining Partners," called *Using Aliases* (on page 263).

Wrapper Window

The Wrapper window allows you to define the structure of a wrapper, also called envelope. Wrappers differ from documents, in that their structure is often that of nested loops, to allow multiple documents within functional groups, and multiple functional groups within interchanges. The Wrapper window reflects this functionality. Wrapper segments are often defined in pairs of headers and trailers, although doing so is not mandatory. In addition, the detail information identifies the header and trailer segments that are part of the document definition.

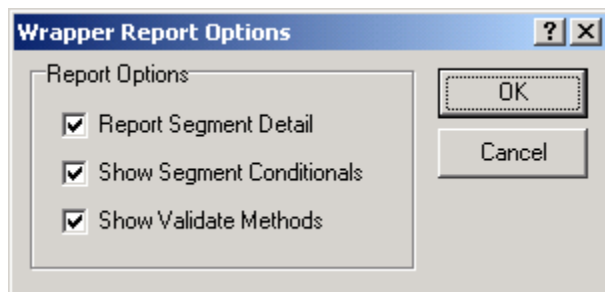
The Wrapper window has an associated *Wrapper Properties* (on page 674) window that defines category, service characters and the type of processing you want to use for input and output.

You also identify user exits here that you have written as C or C++ programs for pre-processing or post-processing of the data. To access the Wrapper Properties window, from the **File** menu, select **Properties**.



Wrapper Window

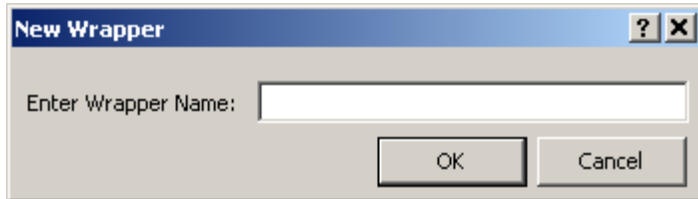
You can print a wrapper report that will show you varying degrees of information. If you choose **Print** or **Print Preview** for a wrapper, the **Wrapper Report Options** dialog box appears.



Wrapper Report Options Dialog Box

Wrapper Name

The wrapper name displays in the title bar. The wrapper name, together with the standard and version, must be unique for all wrapper definitions. It may contain any combination of 1 to 8 alphanumeric characters. You specify the name in the **New Wrapper** dialog box when you *add a wrapper* (on page 667) to a standard version.



New Wrapper Dialog Box

Description (Wrapper)

The description field allows you to enter a textual description up to 80 characters long for the wrapper set.

Blank Column

This narrow column to with the blank header contains a check mark whenever you have attached Edibasic validation routines to this entity. This allows you to see at a glance, for which entities you have written validation routines.

Lvl

The level field indicates at what level the entity functions. The levels may be from 1 through 5, and they must be consecutive. Level 1 is the outermost wrapper and 5 is the innermost. Notice that the header segments and trailer segments have the numbers reversed to identify the header and trailer segments for like levels.

Seq

The sequence number is unique for every segment in the wrapper set. The sequence number may be from 1 to 32,767 (theoretical limit) characters.

Seg Tag

The segment tag is also called the segment ID. It may be 1 to 8 alphanumeric characters.

Description (Segment)

The description field is brought forward from the segment file. Note that it is dimmed.

Rqmt

The requirement designator identifies whether the segment or loop is mandatory (**M**), optional (**O**), or conditional (**C**). If you identify an entity as mandatory, compliance checking generates an error if it does not exist. Optional or conditional entities may or may not be present. Notice that the use of optional and conditional designators varies from standard to standard.

Header (Contents)

The value in this box identifies the document header segment, which is actually defined as the first segment of the document. The segment must exist. In some cases, such as when you define a null wrapper, this segment is a place holder, because the wrapper will not exist in the inbound data. Therefore, when you *define a null wrapper* (on page 146), you can use any segment in this field, because it will not be used.

Trailer (Contents)

The value in this box identifies the optional document trailer segment, which is actually defined as part of the document. It is always the last segment of any document definition for this standard version.

Procedures (Wrapper)

Here are some tasks you may perform from the Wrapper window.

To Add a Wrapper to a Standard Version

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view of the appropriate standard and version subfolders, and choose **Wrappers**.
- 2 From the **File** menu, select **Add**.
The **New Wrapper** dialog box appears.
- 3 Enter the *name* (on page 666) of your new wrapper, and select **OK**.

A blank Wrapper window appears.

- 4 Enter a *description* (on page 666) for your wrapper, and add segments to the wrapper. For instructions, refer to the topic, *To Add a Header or Trailer Segment to a Wrapper* (on page 669).
- 5 After you have added segments to your wrapper, do one of the following:
 - Close the window to save the information
 - or –
 - To add another document:
 - Press **F6**
 - or –
 - Select **Add** from the **File** menu.

The **New Wrapper** dialog box re-appears and you can repeat steps 3-5.

To Modify an Existing Wrapper

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the appropriate standard and version subfolders, and then select **Wrappers**.
- 2 In the right pane, double-click the name of the wrapper you want to modify.
- 3 Make changes and close the window.

The TRM automatically saves the information.

To Save a Wrapper

The TRM automatically saves records when you exit the window, but you may save the configuration at any time.

To make interim saves:

- From the **File** menu, choose **Save**.

To Delete a Wrapper

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the appropriate standard and version to view the subfolders, and then choose **Wrappers**.
- 2 In the right pane, from among the list of wrappers choose the name of the wrapper you want to delete.
- 3 From the **File** menu, select **Delete**.

– or –



From the toolbar, select the **Delete** button

A **Confirm** dialog box appears.

- 4 Select **OK**.

To Sequentially Display Definitions of Wrappers

Sometimes you want to review the wrappers from the Wrapper window. This procedure will display all wrapper definitions from all standards. The standard and version will change when you display definitions from a different standard. From the Wrapper window, to page backward and forward within the definitions:

- 1 To view the next wrapper definition, do one of the following:
From the **View** menu, choose **Next**.
– or –
Press **F7**.
- 2 To view the previous wrapper definition, do one of the following:
From the **View** menu, choose **Prior**.
– or –
Press **F8**.

To Add a Header or Trailer Segment to a Wrapper

Make sure the Data Explorer window is *open* (on page 532).

- 1 If you are already in the Wrapper window, proceed to step 2.
– otherwise –
From Data Explorer, select the wrapper to which you want to add segments.
- 2 Place your cursor in the header or trailer segments detail area.
- 3 Create a blank line.


IMPORTANT: You may insert a line anywhere in the wrapper. They are listed in order based on the unique sequence numbers you use. To insert segments, you must have a gap in your sequence numbers. If you have no gap in your sequence numbers, you must manually resequence other segments to create a gap.

- To add a line at the end, place your cursor in the last line and press the **Tab** key or the down-arrow key.
– or –
- To insert a line before another line, place your cursor on the detail line, and

- Press the **Insert** key
 - or –
- Right-click, and select **Insert Row** from the menu

4 In the blank line, enter the appropriate information, which may include creating a new segment.

NOTE: When you move your cursor to a different line, the Workbench will display the segments in the correct order.

- a) In the **Lvl** (on page 666) column, type the level number.
- b) In the **Seq** (on page 666) column, type the sequence number of the segment.
- c) To add a new segment:
 1. In the **Seg Tag** (on page 667) column, type the segment tag, press **Tab**, and then double-click quickly.
 2. When the Segment window appears, type the appropriate information for the element, and select **OK**.
- d) To select an existing segment:
 1. In the **Seg Tag** (on page 667) column, right-click once.
 2. Click the selection button, , when it appears.
 3. From the **Select Segment** dialog box, select the appropriate element and click **OK**.
- e) In the **Rqmt** (on page 667) column, enter or select the requirement designator.
- f) In the **Contents Header** (on page 667) field, type a segment ID for the header of the document.
- g) Optionally, in the **Contents Trailer** (on page 667) field, type a segment ID for the trailer of the document.

5 Close the window to save your changes.

To Modify a Header or Trailer Wrapper Segment

- 1 From the Wrapper window, position your cursor in the header or trailer segments detail area.
- 2 Use the **Tab** key or the arrow keys to position the cursor to select the field where you want to make the change, and then type the appropriate information.
- 3 Close the window to save your changes.

To Delete a Header or Trailer Segment From a Wrapper

- 1 Place your cursor in the detail area, using the **Tab** key or the mouse.
- 2 If necessary, use the **Tab** key or the arrow keys to position the cursor to the line that you want to delete.
- 3 To delete the line, do one of the following:
 - Press **CTRL+DELETE**.

– or –

- Make sure there is a value in the **Lvl** column, then right-click and select **Delete Row**.

A **Confirm** dialog box appears.

- 4 Select **OK**.

To Generate a Wrapper Text File

When you generate a wrapper, the Workbench includes the definitions of all of its subordinate entities, such as segments and elements in the text file. Therefore, whenever you make changes to the database, you must regenerate the text file, so that you also have the latest changes available for translations. When the Workbench generates a wrapper text file, it creates a backup first if a text file of the same name already exists.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders of the appropriate standard and version, and then select **Wrappers**.
- 2 In the right pane, select the wrapper definition you want to generate.
- 3 From the **Generate** menu, select the name of the wrapper at the bottom of the list.
A progress window appears briefly during generation.

To Add Edibasic Validate Routines to Wrappers

You can add your own validation routines to wrappers, to access the information before mapping. You cannot change the information. For example, such routines allow you to place input information in global variables or to do additional, proprietary compliance checking. The TRM will execute the routine during parsing whenever it encounters this wrapper.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders for the appropriate standard and version, and choose **Wrappers**.
- 2 In the right pane, double-click the desired wrapper.
- 3 Place your cursor in the Wrapper window header area (top part of the window).
- 4 Right-click and select the **Validate** option.
The Edibasic Edit window appears with **Method Validate** header and trailer code.
- 5 Enter your validation routine between the header and trailer code, which typically includes using the **Exception function** (on page 734).
- 6 *Check your syntax for errors* (on page 585).
- 7 Close the window to save your code.
- 8 *Print a wrapper report* (on page 674) to review the validation routines.

To Add Edibasic Validate Routines to Segments within Wrappers

You can add your own validation routines to segments within wrappers. The TRM will execute the routine during parsing only when this segment occurs within this document. The validation routine may override another validation routine written for this segment from the Segment window. You should always review validation routines by printing a document report, so you will know which ones will take effect.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders for the appropriate standard and version, and choose **Wrappers**.
- 2 In the right pane, double-click the desired wrapper.
- 3 Place your cursor in the header or trailer segment detail area on the segment to which you will attach the validation routine.
- 4 Right-click and select the **Validate** option.
The Edibasic Edit window appears with **Method Validate** header and trailer code.
- 5 Enter your validation routine between the header and trailer code, which typically includes using the **Exception function** (on page 734).
- 6 *Check your syntax for errors* (on page 585).
- 7 Close the window to save your code.
The Edibasic Edit window reappears and check mark appears in the blank column, indicating code exists for the segment.
- 8 *Print a document report* (on page 571) to review the validation routines.

To Declare Variables for Validate Routines on Wrappers

You can declare local variables or global variables to use within validation routines for wrappers. You can declare local variables on the **Variables** tab or within the method on the **Validate** tab. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the **Variables** tab only. They have a lifetime from the point of declaration to the end of processing for the input stream. This allows you to store information based on validation routines during the parsing process and use it later to create a proprietary acknowledgment, for example.

Make sure the Data Explorer window is *open* (on page 532).

To create local or global variables for a document, proceed as follows:

- 1 In the left pane, expand the folders to view the subfolders for the appropriate standard and version, and choose **Wrappers**.
- 2 In the right pane, double-click the desired wrapper.
- 3 Place your cursor in the window header area (top part of the window).
- 4 Right-click and select the **Variables** option.

The Edibasic Edit window appears with the **Variables** tab selected.

- 5 Enter your local or global variable declaration statement here.

The following example declares a global variable:

```
Global Gmyvar as Integer
```

The following example declares a local variable:

```
Dim Lmyvar as Integer
```

- 6 **Check your syntax for errors** (on page 585).
- 7 Close the window to save your code.
- 8 **Print a document report** (on page 571) to review the variables and validation routines.

To Declare Variables for Validate Routines on Segments in Wrappers

You can declare local variables or global variables to use within validation routines for segments within a specific wrapper. You can declare local variables on the **Variables** tab or within the method on the **Validate** tab. Local variables declared here have a lifetime and scope of the method. Global variables must be declared on the **Variables** tab only. They have a lifetime from the point of declaration to the end of processing of the input stream. For example, you can store information based on validation routines during the parsing process and use it later to create a proprietary acknowledgment.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders for the appropriate standard and version, and choose **Wrappers**.
- 2 In the right pane, double-click the desired wrapper.
- 3 Position your cursor in the header or trailer segment detail area of the appropriate segment.
- 4 Right-click and choose the **Variables** option.

The Edit window will appear with the **Variables** tab selected.

- 5 Enter your local or global variable declaration statement here.

The following example declares a global variable:

```
Global Gmyvar as Integer
```

The following example declares a local variable:


```
Dim Lmyvar as Integer
```

- 6 **Check your syntax for errors** (on page 585).
- 7 Close the window to save your code.
When you exit the window a check mark appears in the blank column indicating codes exists for the segment.
- 8 **Print a document report** (on page 571) to review the validation routines.

To Print a Wrapper Report

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders for the appropriate standard and version, and choose **Wrappers**.
- 2 In the right pane, choose the wrapper you want to print.
- 3 From the toolbar, select one of the following options, depending on the output:

- The **Print** button  to print reports to a printer or to a file

- The **Print Preview** button  to print reports to the screen

- The **Print to PDF** button  to print to a PDF file

A **Wrapper Report** dialog box appears.

- 4 The default report prints segment and loop information for the document.
 - a) To print element information on the report, select **Report Segment Details**.
 - b) To print segment conditions information on the report, select **Show Segment Conditions**.
 - c) To print the validation methods, select **Show Validate Methods**.
- 5 Select **OK**.

Wrapper Properties Window

The Wrapper Properties window allows you to define processing for any documents using this wrapper definition. It includes defining a new category, if necessary, element code validation for this wrapper only, IO mode, pre-process and post-process methods and service characters for parsing and generation.

You access this window from the Wrapper window. From the **File** menu, select **Properties**.

Procedures (Wrapper Properties)

These are some tasks you can perform from the Wrapper Properties window.

To Specify a Category for a Wrapper

Categories allow you to specify which routine to use for parsing or generating a wrapper. The category is taken from that specified for the standard version, unless you specifically override it for the wrapper. This procedure overrides the category defined for the standard version.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders within the appropriate standard and version, and then select **Wrappers**.
- 2 In the right pane, double-click the wrapper.
The Wrapper window appears.



- 3 From the toolbar, select the **Properties** button.
The Wrapper Properties window appears.
- 4 From the **Category** list, choose the parse and generate method you want to use for this document.
- 5 Select **OK**.

To Check Element ID Codes for a Wrapper

You can invoke element code validation at the standard, document, or wrapper levels. When codes are checked at the standard version level, the status of this wrapper check box has no effect.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders within the appropriate standard and version, and then select **Wrappers**.
- 2 In the right pane, double-click the wrapper whose codes you want to validate.
The Wrapper window appears.
- 3 From the **File** menu, select **Properties**.
The Wrapper Properties window appears.
- 4 Select the **Validate Element Codes** box.
A check mark appears.
- 5 Select **OK**.

To Specify Service Characters for a Delimited Wrapper

You specify which service characters you will use during parse or generate for a particular wrapper and its included documents using the Wrapper Properties window. The wrapper must use delimiters. If it does not, you should not specify any delimiters here.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders for the appropriate standard and version, and then select **Wrappers**.
- 2 In the right pane, double-click the wrapper whose delimiters you want to specify.
- 3 From the **File** menu, select **Properties**.
- 4 Select the **Service Characters** tab, and enter the values required. You may also want to enter an offset. For more information about what to enter, refer to (*Wrapper Properties*) *Service Characters Page* (on page 683).
- 5 Select **OK**.

To Add Default Matching Criteria for Standard Identification

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the appropriate standard and version, and then select **Wrappers**.
- 2 In the right pane, double-click the wrapper to which you want to add matching criteria. The Wrapper window appears.
- 3 From the **File** menu, select **Properties**.
The Wrapper Properties window appears.
- 4 Select the **StdID Match** tab, and then select **New**.
- 5 From the **Operator** list, select the type of operation to be performed on the incoming data to compare it with the given value.
- 6 You can identify both an offset and a value (a) or a combination of segment, field, and subfield together with a value (b). You cannot combine an offset with segment, field or subfield. Proceed as follows:
 - a) In the **Offset** box, enter the number of bytes from the beginning of the file where you want the search to begin.
– or –

- b) In the **Segment** box, enter the absolute sequence of the segment in the incoming data. The TRM has not identified the wrapper definition yet, so it doesn't know how to parse the wrapper segments. If you are identifying a segment tag at the beginning of the segment, this field should be blank.

In the **Field** box, enter the absolute sequence of the element within the segment. If you enter a value in the Segment field, you must enter a value here also.

In the **Sub Field** box, enter the absolute sequence of the component within the composite, if required.

- 7 In the **Value** box, enter the characters against which you will compare the incoming data. These values are also matched for case. If you have embedded spaces, you must enclose the entire value in quotation marks.
- 8 Select **OK**.

To Specify a Pre-Processing User Exit Method for a Wrapper

You may specify a pre-processing method in a user exit for a particular wrapper using the Wrapper Properties window. The method will be executed after the TRM has parsed the wrapper. The user exit DLL file in which this method resides must also be listed on the appropriate list for the target environment. For the Workbench, this list is on the **User Exits** page of the Modify options window. For MessageWay servers on UNIX/Linux and Windows, you maintain the list in the MW Translator Operator program. For more information refer to the *MW Translator Operator Guide and Reference* or the online help for the MW Translator Operator Program.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders within the appropriate standard and version, and then choose **Wrappers**.
- 2 In the right pane, double-click the wrapper.
The Wrapper window appears.
- 3 From the **File** menu, select **Properties**.
The Wrapper Properties window appears.
- 4 Select the **General** tab, and enter the registered name of the method in the **Pre-Process Method** box.
- 5 Select **OK**.

To Specify a Post-Processing User Exit Method for a Wrapper

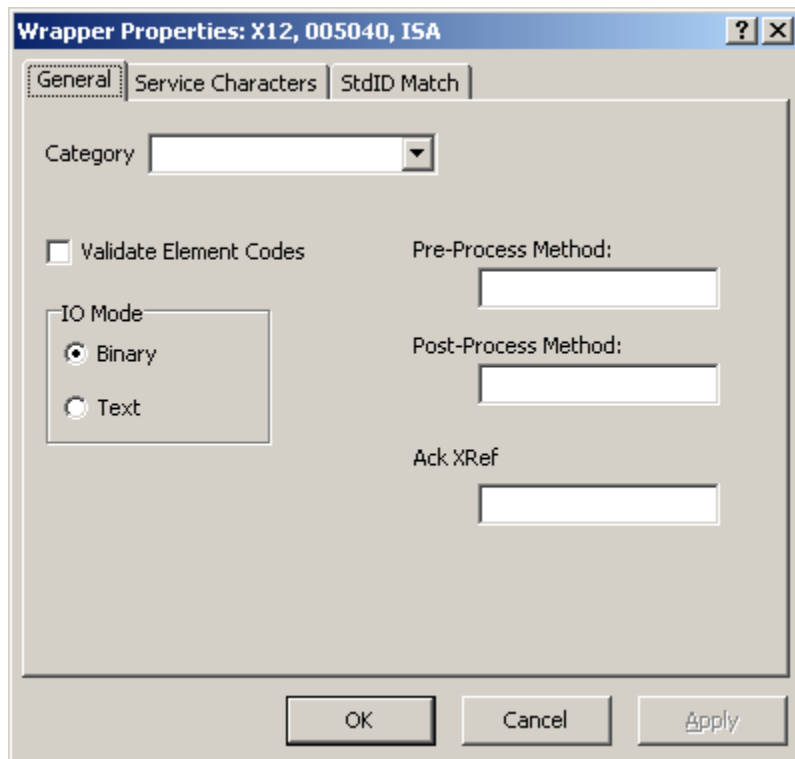
You may specify a pre-processing method in a user exit for a particular wrapper using the Wrapper Properties window. The method will be executed after the TRM has executed the map and before generation of output. The user exit DLL file in which this method resides must also be listed on the appropriate list for the target environment. For the Workbench, this list is on the **User Exits** tab of the Modify Options window. For MessageWay servers on UNIX/Linux and Windows, you maintain the list in the MW Translator Operator program. For more information refer to the *MW Translator Operator Guide and Reference* or the online help for the MW Translator Operator Program.

Make sure the Data Explorer window is *open* (on page 532).

- 1 In the left pane, expand the folders to view the subfolders within the appropriate standard and version, and then choose **Wrappers**.
- 2 In the right pane, double-click the wrapper.
- 3 From the **File** menu, select **Properties**.
- 4 Select the **General** tab, and enter the registered name of the method in the **Post-Process Method** box.
- 5 Select **OK**.

(Wrapper Properties) General

The **General** tab of the Wrapper Properties window allows you to specify a category for parsing and validation, whether or not you want to validate element codes, the parse and generate mode, and a pre- or post-process user exit. If this wrapper requires a cross-reference table to convert values for an acknowledgment, you specify that here.



General Tab (Wrapper Properties Window)

Category

The category identifies the parsing and generation routines that will be used for the standard. You can specify routines from this point down in the structure. When you identify a routine at this level, it will be applied to all lower entities within this standard, if you do not override it with another routine at a lower level.

IMPORTANT: To correctly parse a SWIFT wrapper and document, the **IO Mode** must be **Binary** and the wrappers must use the **{:tag...}** syntax. Refer to Example2 in the subdirectory **\\Workbench\Examples\SWIFT**.

The options available are as follows:

Type	Description
Delimited	Delimited routines use delimiters to parse the data, and generate all data entities separated with delimiters, unless otherwise noted. The delimiters used are the ones defined for the wrapper set that is associated with the interchange at the time of translation.
Fixed	Fixed routines parse the data based on the length of fields. They generate the data by padding to the full length of the field.
SWIFT	This selection invokes SWIFT parsing and generation routines for all lower categories, unless explicitly overridden at a lower level. This is the default if the SWIFT category is selected for a standard. This category implies SWIFT Block processing for the wrapper segments, and SWIFT Block or Fixed processing for the wrapper elements. To correctly parse a SWIFT wrapper and document, the IO Mode must be binary and wrappers must use the {:tag...} syntax. Refer to the example Example2 in the subdirectory \\Workbench\Examples\SWIFT .
XML	Not supported. Define the XML category for the standard on the Standard Version Properties window or for the document on the Document Properties window.

Validate Element Codes

Select this check box to match inbound code values with elements in the wrapper for which codes are defined. When the validation fails, the TRM rejects the interchange. This only works when the **Validate Element Codes** box is not selected on the Standards Version Properties window. When you select or clear the **Validate Element Codes** check box on the Standards Version Properties window, the Workbench also selects or clears this box.

Binary (IO Mode)

This setting determines how the TRM processes input and generates output segments. Whether the TRM uses linefeed, carriage return/linefeed or record breaks for the end of the segment depends on the target environment.

The binary input-output (IO) mode will process all data as a stream by counting the bytes. The input data will be parsed without searching for linefeed, carriage return/linefeed or record breaks as end of segments for fixed-length entities. This means that for fixed-length documents, each record must be filled to its maximum length. This mode ignores any values listed in the **Characters to Ignore Between Segments** field on the Standard Version Properties window. The output will not contain a linefeed, carriage return/linefeed or record breaks at the end of each segment for fixed-format and delimited entities. This option is the default for output on the Workbench environment. You may override this option for output when you run tests on the Workbench by choosing **Force Text** from the **Test** menu.

IMPORTANT: The TRM only uses the **IO Mode** setting from the wrapper definitions of the first interchange in a file.

Text (IO Mode)

This setting determines how the TRM processes input and generates output segments. Whether the TRM uses linefeed, carriage return/linefeed or record breaks for the end of the segment depends on the target environment.

When the **IO Mode** is set to **Text**, The TRM uses either the End of Record (EOR) mark according to the data type for the system on which it is running or the actual byte count, whichever comes first. This allows for short records in fixed-length documents. Once the TRM determines the definition of the first incoming wrapper from the standard identification process, it re-opens the input based on the standard and wrapper definitions. The TRM replaces the carriage-return (CR), linefeed (LF), or record break with a single new-line (NL) character for this and any subsequent interchanges in this file.

Internally, the input data will be parsed using NL as end-of-segment marks for fixed-format entities. The output will contain linefeed, carriage return/linefeed or a record break at the end of each segment for fixed-format and delimited entities.

IMPORTANT: The TRM only uses the **IO Mode** setting from the wrapper definitions of the first interchange in a file. When you have multiple interchanges in a file where the IO Mode is defined as **Text**, and when the segment terminator is the double character, such as, CRLF, you must be careful when you use offset values to identify matching criteria beyond the first segment. You must be sure that the offset for the matching criteria for subsequent interchanges allows for the change in segment terminators from two (CRLF) to one (NL) character.

Pre-Process Method

The TRM only uses the pre-processing user exit associated with the first interchange in a file. The **Pre-Process Method** box allows you to specify the registered name of a method in a user exit for compiled C or C++ functions you have written to pre-process your EDI input. The TRM will execute this routine after it has parsed the input wrapper. It must match the registered name you specify as the **preProcType** option of the **ERMRegisterPreRead** function of your user exit. For additional information, refer to the *MW Translator User Exits Programming Manual*.

Post-Process Method

The **Post-Process Method** box allows you to specify a registered name of a method in a user exit for compiled C or C++ functions you have written to post-process your EDI data. The TRM will execute this routine after it has completed the mapping instructions and before it generates the output. It must match the registered name you specify as the **postProcType** option of the **ERMRegisterPostWrite** function of your user exit. For additional information, refer to the *MW Translator User Exits Programming Manual*.

Ack XRef

The acknowledgment cross-reference box allows you to specify a definition that contains values you want to cross-reference when you generate this wrapper for an acknowledgment. You would create the definition in the **Cross References** folder.

(Wrapper Properties) Service Characters

The **Service Characters** tab allows you to specify either an offset or a value for special characters required for delimited standards. The TRM uses these values to generate delimited standard documents and acknowledgments, unless the acknowledgment profile specifies service characters. During parsing, when an offset is provided, the TRM will use the offset to determine the character value from the incoming data. When there is no offset, the TRM uses the value to parse incoming data.

The screenshot shows a dialog box titled "Wrapper Properties: X12, 005040, ISA". It has three tabs: "General", "Service Characters", and "StdID Match". The "Service Characters" tab is active. It contains a table with two columns: "Value" and "Offset".

	Value	Offset
Segment Terminator	~	105
Tag Delimiter	*	3
Element Delimiter	*	3
Component Delimiter	:	104
Repetition Separator	^	82
Release Character		
Decimal Mark		

At the bottom of the dialog box are three buttons: "OK", "Cancel", and "Apply".

Service Characters Tab (Wrapper Properties Window)

Segment Terminator

The segment terminator is the character used during parsing or generation to mark the end of segments for a delimited entity.

The **Value** box may contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

The **Offset** box contains the byte location, counting the first byte as zero, of the character it is to represent (this can only be used for fixed-length segments, such as, the ISA).

The TRM uses a defined **Offset** character to parse incoming data or the **Value** character when the offset is not defined. When the TRM generates data, it always uses the **Value** character.

Tag Delimiter

The tag delimiter is the character used during parsing or generation to mark the end of a segment tag.

The **Value** box may contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the element delimiter. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

The **Offset** field contains the byte location, counting the first byte as zero, of the character it is to represent (this can only be used for fixed-length segments, such as, the ISA).

The TRM uses a defined **Offset** character to parse incoming data or the **Value** character when the offset is not defined. When the TRM generates data, it always uses the **Value** character.

NOTE: Tag delimiters are not used by X12 or EDIFACT, but are used by TRADACOMS in Europe.

Element Delimiter

The element delimiter is the character used during parsing or generation to mark the end of an element.

The **Value** box may contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the tag delimiter. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

The **Offset** box contains the byte location, counting the first byte as zero, of the character it is to represent (this can only be used for fixed-length segments, such as, the ISA).

The TRM uses a defined **Offset** character to parse incoming data or the **Value** character when the offset is not defined. When the TRM generates data, it always uses the **Value** character.

Component Delimiter

The component delimiter, also called a sub-element separator, is the character used during parsing or generation to mark the end of a component element within a composite.

The **Value** box may contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

The **Offset** box contains the byte location, counting the first byte as zero, of the character it is to represent (this can only be used for fixed-length segments, such as, the ISA).

The TRM uses a defined **Offset** character to parse incoming data or the **Value** character when the offset is not defined. When the TRM generates data, it always uses the **Value** character.

Repetition Separator

The repetition separator is the character used during parsing or generation to distinguish multiple occurrences of a composite or simple element within a segment. This value is used when there is a value or an offset for the Repetition Separator on Wrapper Properties, and when there is a repetition >1 for the composite or the element on the Segment window.

The **Value** box may contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

The **Offset** box contains the byte location, counting the first byte as zero, of the character it is to represent (this can only be used for fixed-length segments, such as, the ISA).

The TRM uses a defined **Offset** character to parse incoming data or the **Value** character when the offset is not defined. When the TRM generates data, it always uses the **Value** character.

Release Character

The release character is the character used during parsing or generation to mark the character that follows it as a true character, rather than one of these service characters, thus releasing the character from its normal duties as an element separator, for instance.

The **Value** box may contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

The **Offset** box contains the byte location, counting the first byte as zero, of the character it is to represent (this can only be used for fixed-length segments, such as, the ISA).

The TRM uses a defined **Offset** character to parse incoming data or the **Value** character when the offset is not defined. When the TRM generates data, it always uses the **Value** character.

Decimal Mark

The decimal mark is the character used during parsing or generation in numeric values that use an explicit decimal.

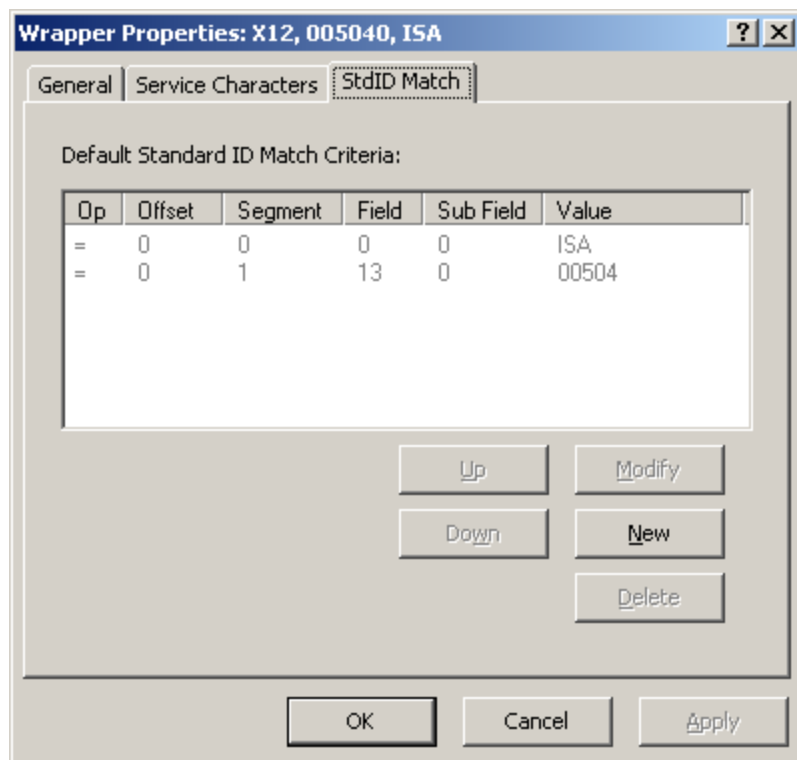
The **Value** box may contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

The **Offset** box contains the byte location, counting the first byte as zero, of the character it is to represent (this can only be used for fixed-length segments, such as, the ISA).

The TRM uses a defined **Offset** character to parse incoming data or the **Value** character when the offset is not defined. When the TRM generates data, it always uses the **Value** character.

(Wrapper Properties) StdID Match

The **StdID Match** tab allows you to specify default criteria that can be used during standard identification to select a wrapper definition to parse the input data.



StdID Match Tab (Wrapper Properties Window)

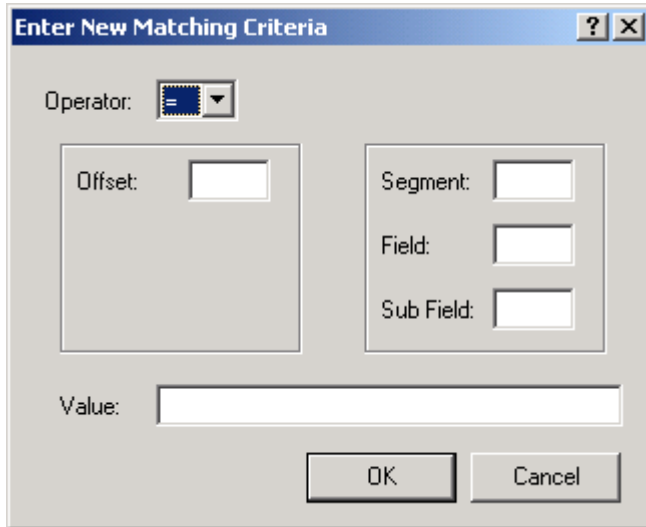
When you choose the **Modify** or **New** button, the **Enter New Matching Criteria** dialog box appears.

The dialog box allows you to further qualify matching of a particular wrapper among several possibilities for a location. If there is only one wrapper ever associated with the location, there is no need to use the matching data fields. You may use one of two criteria:

- Offset position
- or –

- Segment, field and sub-field positions

The matching value must occur within the first buffer of data, typically 1k-4k, depending on the system.

The image shows a dialog box titled "Enter New Matching Criteria". It has a standard Windows-style title bar with a question mark icon and a close button. The dialog contains several input fields: "Operator" with a dropdown menu showing an equals sign; "Offset" with a text box; "Segment" with a text box; "Field" with a text box; "Sub Field" with a text box; and "Value" with a large text box. At the bottom, there are "OK" and "Cancel" buttons.

Enter New Matching Criteria Dialog Box

Default Standard ID Match Criteria

This default matching criteria transfers to the **General** tab of the Standard ID window when you add a wrapper to a location in Partner Explorer. The criteria are used to match a particular wrapper among several possibilities. If there is only one wrapper associated with the location, there is no need to use the matching data fields. The definitions for the matched wrapper are then used to parse the input wrapper data. The matching value must occur within the first buffer of data, typically 1k-4k, depending on the system. Some default matching criteria are already entered for public standard wrappers.

IMPORTANT: To identify the incoming standard, the TRM reads the input file in raw mode as binary data. When it identifies the standard, it then re-opens the file. Then any required pre-processing user exit performs its tasks. Otherwise, when the **IO Mode** is **Text**, the TRM replaces line representations such as carriage-return/line-feed (CRLF), line-feed (LF) or record mark characters with an internal new-line (NL) character. When there are multiple interchanges within the input file, subsequent standards must be identified with matching criteria that allow for a possible change in the number of characters when using offset values.

Operator (Match Criteria)

You use one of these operators to compare the incoming value with the value given for this condition. The operator symbol identifies the type of condition that is applied, as follows:

Operator	Function
=	equal
<>	not equal
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal

Offset (Match Criteria)

The offset determines location from the beginning of the file where you begin counting. Zero (0) offset begins comparing with the first byte, which for delimited standards is the beginning of the segment tag. If you enter an offset, you cannot use segment, field, or sub-field.

Segment (Match Criteria)

The segment is the absolute sequence number of the segment that contains the incoming data for comparison. If you choose to match by segment, field, and potentially sub-field, you cannot enter an offset. If you enter a value for **Segment**, you must also enter a value for **Field**.

Field (Match Criteria)

The field corresponds to the element or composite element of the segment that contains the incoming data for comparison. It should contain the absolute number of the field, beginning with the first field, typically the segment tag, as field number 1. If you choose to match by segment, field, and potentially sub-field, you cannot enter an offset. If you enter a value for **Field**, you must also enter a value for **Segment**.

SubField (Match Criteria)

If the **Field** identifies a composite element, this identifies the component within the composite. The number of the first component element is field number 1. If you choose to match by segment, field, and potentially sub-field, you cannot enter an offset.

Value (Match Criteria)

This value is the string against which the input will be compared. Valid characters include any ASCII printable characters and hex values in the form, `\xnn`, where `nn` is the hex value, for example, `\x20` for a space. For EBCDIC data, you must place an **E** before the value, and the value must be enclosed in quotation marks, such as `E"HDR"`.

Trailing spaces entered for the value are deleted. To maintain trailing spaces to match an input value with trailing spaces, you may use a hex value at the end of the value field. For example, for an element of 15 characters, where the input value is **SENDPARTNER** followed by 4 spaces, in the value field of the **Enter New Matching Criteria** dialog box you would enter **SENDPARTNER** followed by three spaces and terminate with a hex space.

Up Button

The **Up** button allows you to move the selected criteria up one position in the matching order. The TRM terminates the matching process when it encounters a value that does not match the incoming data.

Down Button

The **Down** button allows you to move the selected criteria down one position in the matching order. The TRM terminates the matching process when it encounters a value that does not match the incoming data.

Modify Button

The **Modify** Button allows you to add matching criteria to match with the incoming data. This allows you to make distinctions between various definitions based on incoming wrapper data.

New Button

The **New** button allows you to enter new matching criteria.

Delete Button

The **Delete** button allows you to delete matching criteria.

This page intentionally blank

Edibasic Reference

Introduction to Edibasic

Edibasic is the language provided for use during parsing for user validation or for use during mapping to augment visual mapping instructions. You would use Edibasic if you want to add specific validation routines not part of the default compliance checking. You would also use it when your map requires a more complex manipulation of data than you can achieve with simple relationships of input and output.

You can create various methods to control user validation and the mapping of loops, segments, or elements. Methods used during validation are called during parsing before any routing or translation is done. Methods used during mapping are similar to subroutines, in that you write them, and the translator then calls them during generation of the output. Within these methods, you can use built-in functions, and predefined constants to achieve your desired result. You can also define and manipulate variables and access data elements from the incoming wrapper or document.

Edibasic Syntax Requirements

The following conventions should be observed when using the mapping language:

- The mapping language is not case sensitive. Keywords, segment IDs, and variables may be used in any combination of upper and lower case.
- String literals must be between single or double quotes and are case-sensitive.
- Comments begin with REM or an apostrophe (') and end at the end of the line, and have a maximum length of 256 characters.
- Variable names must not begin with REM (I.E. remotefile is invalid).
- The limit on a code page in the Edibasic Editor is 32K.

Keywords

The following keywords are reserved and may not be used as variable or constant names.

a_abort	a_accept	a_reject	abs
and	as	asc	avg
bcd	case	chr\$	const
convert\$	count	dateserial	day

dim	do	doc	each
else	elseif	end	eqv
err	exit	f_ack_control_ref	f_ack_count
f_ack_expected	f_ack_request	f_ack_status	f_ack_subject_docid
f_agency	f_assoc	f_comp_delim	f_control_ref
f_count	f_date	f_decimal_mark	f_doc_id
f_doc_version	f_ele_delim	f_func_grp_id	f_ich_version
f_password	f_password_qual	f_rec_id	f_rec_intid
f_rec_intid2	f_rec_mailbox	f_rec_qual	f_rec_subid
f_record_tag	f_release	f_release_char	f_repeat_sep
f_seg_term	f_send_id	f_send_intid	f_send_intid2
f_send_mailbox	f_send_qual	f_send_subid	f_set_id
f_tag_delim	f_test_ind	f_time	f_user_field1
f_user_field2	f_user_field3	f_user_field4	f_val_field1
f_val_field2	f_val_field3	f_val_field4	field\$
for	format\$	global	hour
if	imp	initocc	instr
integer	is	isnull	isnumeric
lcase\$	left\$	len	let
loop	ltrim\$	max	method
mid\$	min	minute	mod
month	next	nextocc	not
now	off	on	or
print	rem	resetocc	right\$
rtrim\$	second	segment	select
setfield	space\$	ssedoc	ssewrap
step	str\$	strcomp	string
string\$	sum	then	timeserial
to	trace	trim\$	ucase\$
until	user	val	value
variant	vartype	vtbcd	vtdate
vtinteger	vtnull	vtstring	weekday

wend	while	wrap	xor
xref\$	xrefr\$	year	

Variable Data Types

Data types used within Edibasic are distinct from the standard data types defined for elements. Data types assigned to elements within a standard control data compression and padding and data formatting for numbers and datetime values when using visual mapping.

The data types used with Edibasic variables are integer, BCD, string and variant.

The default data type for variables is variant. You can specifically declare a data type using **Method Start**. You can optionally declare the type by adding one of the type designators to the end of the name:

- \$ string
- % integer
- @ BCD

Integer

Integers are signed numeric values between the range of -2,147,483,648 and 2,147,483,647, inclusive.

BCD

Binary-coded decimal (BCD) variables are floating decimals with up to 18 digits of precision.

String

Strings are arbitrary byte sequences from 0 to 65,535 bytes in length. Characters within strings may contain any byte value (0-255).

Variant

This is the default data type for a temporary variable. Variant types can contain integer, bcd, string, or date values. Variant dates store a time and/or date in an internal format. You initialize variant dates using the `Now` function or the **Value** function with a valid date or time format.

Variable Names

Variable names have the following requirements:

- Begin with an alpha character (letter).
- Maximum of 40 characters long.
- May contain letters, numbers, and the underscore (_).
- Must not begin with REM (I.E. remotefile is invalid).
- May add a character to the end of the name to declare the data type, rather than declare it in a **Method Start**, as follows:
 - \$ string
 - % integer
 - @ BCD

IMPORTANT: Remember that variable names are NOT case-sensitive.

Variable Types and Declaration

To store values temporarily, you may create three types of variables: global, local, and temporary (declared and undeclared). You explicitly declare global variables and local variables. You may also declare temporary variables. Edibasic creates temporary, undeclared variables for any names that it does not recognize as global, local or declared temporary (variable names are not case-sensitive). Global, local and declared temporary variables may be any of the following types: integer, bcd, string, or variant. Undeclared temporary variables default to the variant type.

For use with input documents, variables may be declared within a user validation routine for a document, wrapper, loop/group, segment, composite element, or element definitions. For use during mapping, variables may be declared at the document or wrapper level, the loop/group level, the segment level, the element (or composite element) level or the sub-element (component element) level.

Variables are created as follows:

- *Global* variables must be declared on a **Variables** tab using the keyword, **Global**
- *Local* variables must be declared on a **Variables** tab using the keyword, **Dim**
- *Temporary* variables may be undeclared, otherwise must be declared on a **Method** tab using the keyword, **Dim**

IMPORTANT: You may use a maximum of 32,768 variables per **Variables** page or within a method.

Variable Lifetime and Scope

Variables have both a lifetime and a scope. The lifetime of a variable is as long as it remains in memory, until the memory area is re-initialized. The TRM re-initializes memory as follows:

- For *global* variables, the lifetime is between input streams. Global variables are created when first referenced and destroyed when processing of the input stream ends. A global variable, whether it is created in the input document within a validation method or in the map, persists across all output, such as, documents, wrappers and acknowledgments, that are created from the same input stream
- For *local* variables, the lifetime is between documents in an input stream, when the next document is different from the previous one. Otherwise, the value stays in memory until the user initializes it, usually during a **Method Start** routine. Local variables are created when referenced.
- For *temporary* variables, declared or undeclared, the lifetime is for the current method of the current entity. These variables are created when they are referenced within the method, and they are destroyed when the method ends.

Scope is object-specific. The scope of a variable determines whether you can access the variable. The scope of a variable is that object, including its children. For example, if you declare a variable on a **Variables** tab for a loop/group, the scope of the variable is that loop/group: it is accessible from any method defined for that loop/group or its segments, nested loops/groups, and elements. If you declare a local variable within a method or the TRM creates a temporary variable, the scope of the variable lasts until the termination of the method.

Lifetime and scope together determine whether you can access a given variable. The type of variable created determines its lifetime. When you declare a global or local variable, you set the scope, by virtue of the object to which you attach the function statement. If you want to access a variable outside of the scope you set, you must declare the variable again for the new scope. For example, if you declare a global variable, GVAR, for a segment, ABC, within a user validation routine, the lifetime of the variable is the entire input stream and the scope of the variable is that object, segment ABC, in the input document. Later, within the processing of that same input stream, you want to use that variable to create data for a backward acknowledgment. To access that variable, which still exists because processing of the input stream is not yet complete, you must set the scope by declaring the GVAR variable again, this time for a new object, segment XYZ, which is a segment within the acknowledgment.

NOTE: As a basic rule, you should always initialize variables before you use them. You create variables using the **Variables** tab on the Methods Editor window. You initialize variables using *Method Start*.

Literals

Literals are values used as an operand. They will be assigned one of three data types: integer, BCD or string.

Integer

Integers are signed numeric values between the range of -2,147,483,648 and 2,147,483,647, inclusive. Note that values beyond the range will be treated as BCD data types, even though they contain no decimal mark.

IMPORTANT: When you assign a BCD value to an integer variable, the integer value is determined MOD 2147483648. When the BCD value is greater than 2147483647, you will not get an overflow error. MW Translator does not provide for overflow checking.

BCD

Binary-coded decimals (BCD) literals are floating decimals with up to 18 digits of precision. Integer values (no decimal mark) beyond the integer range are also treated as BCD literals.

String

String literals start and end with single or double quotes. Strings are arbitrary byte sequences from 0 to 65,535 bytes in length.

TIP: The length of a string literal is limited by line length, which is currently 255 characters. If you want to use a string literal that is longer than 255 characters, you can use the concatenation operand. Characters within strings may contain any byte value (0-255).

Element Data Types

All elements are treated as strings. If you need to change the data in a way that is not possible with visual mapping, you may use Edibasic routines and variables. With Edibasic routines, you can manipulate data not only as strings, but also as integer, BCD, and variant (includes datetime) data types.

Data Element Names

Data element names are used to retrieve information from the data element store for the input document, the input wrapper, or for status and error information collected during processing. The data element name contains at least a segment tag and an element sequence number, separated with a period. It may contain much more, depending on the current scope and what you need to access outside of this scope.

There may be times when you want to access particular instances of information, where the current scope would not allow you to do so. You can do so if you understand how to access the information, part of which depends on using optional parts of the element name. For more information about these mapping techniques, refer to the topic *Advanced Mapping Techniques* (on page 185).

IMPORTANT: The segment tag portion of the element name must comply with the rules for variable names. If it does not, perhaps because it begins with a number, you must enclose the segment tag in square brackets, [].

The full syntax for the element name is as follows

```
{des.} seg-tag|[seg-tag]{[area,]seg-seq-no}{occ,occ...}.ele(occ):comp-ele
```

des optional

The data element store (des) is one of the locations where you can retrieve input and status, statistic, and error information that you can use during mapping. The four locations are:

- **DOC** contains the input document data (default for document sets)
- **WRAP** contains the input wrapper data (parsed to this point) (default for wrapper sets)
- **SSEDOC** contains the status, statistic, and error information for the document collected to this point
- **SSEWRAP** contains the status, statistic, and error information for the wrapper(s) collected to this point.

For example, if you want to reference the WRAP des, you would enter the following:

WRAP.

seg-tag | [*seg-tag*] mandatory

The segment tag is the segment ID in the standards definition to which the element belongs. Normally, the segment tag used in Edibasic statements is the same as the segment ID.

The following example references the N1 segment:

N1

However, when the segment ID does not conform to the requirements of **variable names** (on page 696), you must place brackets ([]) around the segment tag in any Edibasic statements. For example, if your segment ID begins with a number, rather than a letter, e.g., **010**, it must be referenced as follows:

[010]

area optional

If you need to distinguish between identical segment tags within different areas, you must use the area reference as given in your standards definitions for the document. When you use the area, you must also use the segment sequence number (refer to following description). The area is numeric, and is typically used as follows:

- 1 header information
- 2 detail information
- 3 summary information

The following example references the N1 segment in the header area (1), rather than the detail area (2). The reference includes the area, the obligatory comma and segment sequence number (310) all enclosed in braces following the segment tag:

N1{1,310}

seg-seq-no optional

If you need to distinguish between identical segment tags within a document, you must use the segment sequence number and enclose it in braces. If the tag and sequence number are not unique within a document, then you must also use the area number and both must be enclosed in the braces and separated by a comma (refer to preceding description).

When you are sure that a segment sequence number is not reused within the document, you may omit the area, using only the segment sequence number in braces:

REF{580}

(occ,occ...) optional

If you need to access a specific occurrence or occurrences of a repeating segment or a loop, you can do so using a valid expression, which must be enclosed within parentheses. The expression should evaluate to an integer value ≥ 1 . To identify multiple occurrences, you must separate them with commas. If there are fewer expressions than the nesting level of the segment, then the expressions will be right-aligned. That is, the last expression will correspond to the innermost loop or segment repetition.

The following example refers to the third occurrence of the N4 segment within the current scope.

N4(3)

This next example refers to the second occurrence of the N4 segment within the third occurrence of the N1 loop within the current scope.

N4(3,2)

.ele mandatory

Elements are identified by the element sequence number assigned to them in the standards definition for the segment. Element sequence numbers are assigned to simple and composite elements. There may be gaps in the numbers, but the numbers determine the order of elements in the segments. It is separated from the preceding segment information by a period.

The following example refers to the first element in the N1 segment:

N1.1

(occ) optional

If you need to access a specific occurrence or occurrences of a composite or simple element, you can do so using a valid expression, which must be enclosed within parentheses. Component elements do not repeat. The expression should evaluate to an integer value ≥ 1 .

The following example references the third occurrence of the first element in the COM segment.

COM.1(3)

:comp-ele: mandatory for component element

The component element number is the sequence number assigned to it in the standards definition for the element. It is separated from the preceding element number by a colon (:). If you are accessing a component element within a composite, you must identify the composite element number and the sequence number of the component element within the composite. The sequence number of the composite element is the element sequence number described above.

The following example references the first component in the first element, which is a composite, in the DTM segment:

DTM.1:1

Examples

The following are a few examples. You will find additional examples of complete element names in map reports.

- This is the simplest, and most common form of the data element name, using the element sequence number (01) and the segment tag (BEG):

BEG.01

- This example shows the data element name for the control reference element, which is 1, contained in the status, statistics, and error location for the incoming wrapper:

SSEWRAP.WRAP.1

- This example references an element where there are multiple ITA segments: 4 in table 2 (detail area), and 2 in table 3 (summary area). Of the one in table 2, which has a segment sequence number of 180, you want element sequence number 7 from the first occurrence of this ITA.

ITA{2,180}(1).7

- This example references a composite element contained within a COM segment in table 2 (detail area), which has a segment sequence number of 500. You want the second occurrence of element sequence number 3, which is a repeating composite element, from the first occurrence of this COM segment.

COM{2,500}(1).3(2)

- This example references a simple element contained within a COB segment in table 2 (detail area), which has a segment sequence number of 4000. You want the 3rd occurrence of a repeating element, sequence number 4, from the current occurrence of this COB segment, which actually only occurs once.

COB{2,4000}.4(3)

- This example shows the use of brackets when you use an element that is in a segment that uses numeric values for its segment tag. This represents data element sequence number 1, of segment sequence number 6, which is in table 1 (header area) and whose segment tag is 201.

[201]{1,6}.1

Operators

Operators identify processes, such as mathematical calculations, concatenations, or pattern matching, that result in a value. Operators perform a function on one or more operands. There are three types of operators: primary, unary, and binary.

Type	Operators
Primary	() {}
Unary	- Not
Binary	+ - * / \ Mod & = <> < > >= <= And Or Xor Eqv Imp

Expressions and Operators

Operators are only useful when they occur in an expression. An expression is any combination of element names, variable names, constants, literals, or a combination of sub expressions and operators that produces a value.

Primary Expressions and Operators

Primary expressions include previously declared elements, variables, constants, or literals. The functions of those used in Edibasic are explained in the following table.

Operator	Description
()	Changes associativity of operators or Encloses an occurrence expression
{ }	Encloses an argument list Encloses a level and segment sequence expression

Unary Expressions and Operators

Unary expressions are formed by combining a unary operator with a single operand. The functions of those used in Edibasic are explained in the following table.

Operator	Description
-	Negates an arithmetic expression. The result is the negative of the expression.
NOT	Negates an expression. The result is the negative of the expression: false if the expression is true, or true if the expression is false.

Binary Expressions and Operators

Binary expressions contain a binary operator and two operands. The functions of those used in Edibasic are explained in the following table.

Function	Operator	Description
Additive	+	Adds two operands.
	-	Subtracts two operands.
Multiplicative	*	Multiplies two operands
	/	Divides two operands and returns a floating point result.
	\	Divides two operands and returns an integer result.
	Mod	Divides two operands and returns only the remainder.
Concatenating	&	Concatenates two operands
Relational	=	Equality
	<>	Inequality
	<	Less than
	>	Greater than
	<=	Less than or equal to
	>=	Greater than or equal to
Logical	And	within a conditional statement, performs a logical conjunction on two expressions, and returns a result of true or false.
	Or	within a conditional statement, performs a logical disjunction on two expressions, and returns a result of true or false.

Function	Operator	Description
	Xor	within a conditional statement, performs a logical exclusion on two expressions, and returns a result of true or false.
	Eqv	within a conditional statement, performs a logical equivalence on two expressions, and returns a result of true or false.
	Imp	within a conditional statement, performs a logical implication on two expressions, and returns a result of true or false.

Operator Syntax

The following reference material is organized by operator symbols.

Not operator

Description	within a conditional statement, performs a logical negation on an expression, and returns a result of true or false.
Syntax	Not <i>expression</i>
Valid Values	<i>expression</i> any numeric variable, constant, data-element or function
Notes	If the expression is true, the returned result is false. If the expression is false, the returned result is true.
See Also	<i>Unary Expressions and Operators</i> (on page 705)

+ operator

Description	adds two numbers.
Syntax	<i>result</i> = <i>operand1</i> + <i>operand2</i>
Valid Values	<i>operand</i> any variable, constant, data-element or function that contains data of types integer, bcd, or variant.

- operator

Description	<ul style="list-style-type: none">subtracts one number from another (as type 1, binary operator) – or –produces the negative value of a number (as type 2, unary operator).
Syntax	type1 $result = operand1 - operand2$ type2 $-operand$
Valid Values	<i>operand</i> any variable, constant, data-element or function that contains data of types integer, bcd, or variant.
See Also	Not operator (on page 706)

* operator

Description	multiplies two numbers.
Syntax	$result = operand1 * operand2$
Valid Values	<i>operand</i> any variable, constant, data-element or function that contains data of types integer, bcd, or variant.

/ operator

Description	divides two numbers and returns a floating-point number.
Syntax	$result = operand1 / operand2$
Valid Values	<i>operand</i> any variable, constant, data-element or function that contains data of types integer, bcd, or variant.

\ operator

Description	divides two numbers and returns only the integer (no remainder).
Syntax	$result = operand1 \setminus operand2$
Valid Values	<i>operand</i> any variable, constant, data-element or function that contains data of types integer, bcd, or variant.

Mod operator

Description	divides two numbers and returns only the remainder (no integer).
Syntax	<i>result = operand1 Mod operand2</i>
Valid Values	<i>operand</i> any variable, constant, data-element or function that contains data of types integer, bcd, or variant.

& operator

Description	concatenates two strings.
Syntax	<i>result = operand1 & operand2</i>
Valid Values	<i>operand</i> any variable, constant, data-element or function that contains string data.

= operator

Description	within a conditional statement, compares two expressions for equality and returns a result of true or false.
Syntax	<i>expression1 = expression2</i>
Valid Values	<i>expression</i> any variable, constant, data-element or function that can be used in a string or numeric comparison
Notes	returns true if <i>expression1</i> is equal to <i>expression2</i> , false if <i>expression1</i> is not equal to <i>expression2</i> .

<> operator

Description	within a conditional statement, compares two expressions for inequality and returns a result of true or false.
Syntax	<i>expression1 <> expression2</i>
Valid Values	<i>expression</i> any variable, constant, data-element or function that can be used in a string or numeric comparison.
Notes	returns true if <i>expression1</i> is not equal to <i>expression2</i> , false if <i>expression1</i> is equal to <i>expression2</i> .

< operator

Description	within a conditional statement, compares two expressions to see if the first is less than the second, and returns a result of true or false.
Syntax	<i>expression1 < expression2</i>
Valid Values	<i>expression</i> any variable, constant, data-element or function that can be used in a string or numeric comparison.
Notes	returns true if <i>expression1</i> is less than <i>expression2</i> , false if <i>expression1</i> is greater than or equal to <i>expression2</i> .

> operator

Description	within a conditional statement, compares two expressions to see if the first is greater than the second, and returns a result of true or false.
Syntax	<i>expression1 > expression2</i>
Valid Values	<i>expression</i> any variable, constant, data-element or function that can be used in a string or numeric comparison.
Notes	returns true if <i>expression1</i> is greater than <i>expression2</i> , false if <i>expression1</i> is less than or equal to <i>expression2</i> .

<= operator

Description	within a conditional statement, compares two expressions to see if the first is less than or equal to the second, and returns a result of true or false.
Syntax	<i>expression1 <= expression2</i>
Valid Values	<i>expression</i> any variable, constant, data-element or function that can be used in a string or numeric comparison.
Notes	returns true if <i>expression1</i> is less than or equal to <i>expression2</i> , false if <i>expression1</i> is greater than <i>expression2</i> .

>= operator

Description	within a conditional statement, compares two expressions to see if the first is greater than or equal to the second, and returns a result of true or false.
Syntax	<i>expression1 >= expression2</i>

Valid Values	<i>expression</i> any variable, constant, data-element or function that can be used in a string or numeric comparison.
Notes	returns true if expression1 is greater than or equal to expression2, false if expression1 is less than expression2.

And operator

Description	within a conditional statement, performs a logical conjunction on two expressions, and returns a result of true or false.
Syntax	<i>expression1 And expression2</i>
Valid Values	<i>expression</i> any numeric variable, constant, data-element or function.
Notes	Returns true if both of its expressions are true, otherwise it returns false. If one of the expressions is null and the other is not false, then the result is null.
See Also	Or (on page 710), Xor (on page 710), Eqv (on page 711), Imp (on page 711)

Or operator

Description	within a conditional statement, performs a logical disjunction on two expressions, and returns a result of true or false.
Syntax	<i>expression1 Or expression2</i>
Valid Values	<i>expression</i> any numeric variable, constant, data-element or function.
Notes	Returns true if either of its expressions is true, otherwise it returns false. If one of the expressions is null and the other is not true, then the result is null.
See Also	And (on page 710), Xor (on page 710), Eqv (on page 711), Imp (on page 711)

Xor operator

Description	within a conditional statement, performs a logical exclusion on two expressions, and returns a result of true or false.
Syntax	<i>expression1 Xor expression2</i>
Valid Values	<i>expression</i> any numeric variable, constant, data-element or function.

Notes Returns true if only one of the expressions is true and the other is not null. Returns null if either expression is null. Returns false if both expressions are either true or false.

See Also **And** (on page 710), **Or** (on page 710), **Eqv** (on page 711), **Imp** (on page 711)

Eqv operator

Description within a conditional statement, performs a logical equivalence on two expressions, and returns a result of true or false.

Syntax *expression1 Eqv expression2*

Valid Values *expression*

any numeric variable, constant, data-element or function.

Notes Returns true only when both expressions are true or false. Returns false when only one of the expressions is false and the other is not null. Returns null if either expression is null.

See Also **And** (on page 710), **Or** (on page 710), **Xor** (on page 710), **Imp** (on page 711)

Imp operator

Description within a conditional statement, performs a logical implication on two expressions, and returns a result of true or false.

Syntax *expression1 Imp expression2*

Valid Values *expression*

any numeric variable, constant, data-element or function.

Notes Returns true if both expressions are true or both are false or if the second expression is true, or if the first expression is false and the second is null. Returns false only when the first expression is true and the second is false. Otherwise, it returns null.

See Also **And** (on page 710), **Or** (on page 710), **Xor** (on page 710), **Eqv** (on page 711)

Method Syntax

You can supplement visual mapping by writing Edibasic code for one or more of the following predefined methods. You can define these methods at the document or wrapper, loop, segment, or element level. Once defined, the translator automatically calls them during the generation of that part of the output. Here we define the syntax for these methods.

IMPORTANT: Edibasic automatically inserts the method header (first line) and method trailer (last line). DO NOT CHANGE THESE LINES.

Method Condition as Integer

Description	Returns a numeric value indicating whether a loop or segment may be generated: Zero does not allow entity to be generated; non-zero does.
Syntax	Method Condition as Integer [<i>statements</i>] Condition = <i>expression</i> End Method
Valid Values	<i>statements</i> any valid Edibasic statements <i>expression</i> entity that produces a value upon which the Condition function acts.
Notes	Use this method for documents, repeating or non-repeating loops or segments, or elements to specify when to generate the entity to which the condition applies. The TRM tests the condition before executing mapping instructions related to the entity. If the method returns a zero, then the document, loop, segment, or element is not generated, otherwise it is generated. That is, if the method returns a value other than zero, the TRM will attempt to generate the entity by applying applicable mapping instructions.
Example	

The following example shows a condition to be attached to an outbound repeating segment (repeat number in variable `Idx2`) within a loop (repeat number in variable `Idx`), which will keep you from generating blank output segments if you have no input, which you might need to do if you are generating the output using Edibasic mapping rather than visual mapping. The variable `Idx` would be declared on the N1 loop and `Idx2` would be declared on the PER segment (code not shown here).

Method Condition As Integer

```

Condition = 1           'initialize value to more occ.
Select Case Idx
  case 1               'if 1st occ. of N1
    Select Case Idx2
      case 1          'if 1st occ. of PER
        If BS1.7 = '' then 'if no data
          Condition = 0
        End If
        Exit Method
      case 2          'if 2nd occ. of PER
        If BS1.11 = '' then
          Condition = 0
        End If
        Exit Method
    End Select
  End Select

  case 2               'if 2nd occ. of N1
    Select Case Idx2
      case 1
        If BS1.18 = '' then
          Condition = 0
        End If
        Exit Method
    End Select
End Select
End Method

```

Method GetNext as Integer

Description Returns a value indicating whether another occurrence of a loop or segment is to be generated. This method generates loops and segments, unless it is overridden by a condition.

Syntax

```

Method GetNext as Integer
    [statements]
    GetNext = expression
End Method

```

Valid Values	<i>statements</i>	any valid Edibasic statements
	<i>expression</i>	entity that produces a value upon which the GetNext function acts.
Notes	Use this method for potentially repeating loops or segments to determine if there is an occurrence to process. Executes before generating each occurrence of a given loop or segment. If the method returns a zero, neither the next occurrence nor any additional occurrences are generated. If the method returns any other value, usually 1, then the current repetition of the loop or segment is generated, after which the method is called again for the next occurrence. Generation of a particular instance can be overridden by Method Condition .	
Warning	You must make certain that GetNext eventually returns a zero (0), or your translation will be caught in an interminable loop.	

Example

The following example uses the function, **NextOcc** to determine if there are any more occurrences of the N1 loop. The result is stored in a variable called Rslt, which is then passed to **GetNext**. This could also be accomplished by drag-and-drop mapping of the N1 loop.

```
Method GetNext as Integer
    Rslt = NextOcc(N1)
    GetNext = Rslt
End Method
```

Method MapEle as String

Description	Returns a string value for the element based on whatever means were used to create the data using Edibasic. This method generates elements.	
Syntax	Method MapEle as String [<i>statements</i>] MapEle = <i>expression</i> End Method	
Valid Values	<i>statements</i>	any valid Edibasic statements
	<i>expression</i>	entity that produces the string value for the output element.
Notes	Use this method to generate elements. <i>Expression</i> must evaluate to a string value, which will be the element value that is generated. Elements will be generated only when the value is not null. You must use this method rather than the Literals button on the mapping window for literals that are longer than 43 characters.	
Example		

The following example shows how you can move the literal "element value" into an element.

```
Method MapEle as String
    MapEle = "element value"
End Method
```

Method Start

Description	Initializes loops or segments before generating a loop or segment. Also initializes variables or calculates values used later to generate a composite element or document.
Syntax	Method Start [<i>statements</i>] End Method
Valid Values	<i>statements</i> any valid Edibasic statements
Notes	Typically used to initialize occurrences for repeating loops or segments, but can also be used for non-repeating loops or segments. It is also used for composite elements and for the entire output document.

Example

The following example uses the function, **InitOcc** (on page 741), to initialize the data element store's occurrences of the N1 loop. The result is passed to Rslt. This could also be accomplished by drag-and-drop mapping of the N1 loop.

```
Method Start
    Rslt = InitOcc(N1)
End Method
```

Method Stop

Description	Resets the nested repetition level of loops or segments.
Syntax	Method Stop [<i>statements</i>] End Method
Valid Values	<i>statements</i> any valid Edibasic statements
Notes	Executes after all repetitions of the loop or segment have been generated. It is typically used for nested repeating loops or segments to reinitialize repetitions.

Example

The following example uses the function, **ResetOcc** (on page 749), to reset the data element store's occurrences of the N1 loop to what they were before the last call to the function, **InitOcc** (on page 741). The result is passed to Rslt. This could also be accomplished by drag-and-drop mapping of the

N1 loop.

```
Method Stop
    Rslt = ResetOcc (N1)
End Method
```

Method Validate as Integer

Description Allows users access to Edibasic during parsing, in order to create custom user-validation of the incoming wrapper and/or document data. If the entity to be validated repeats (loop or segment), the method is executed after each occurrence is parsed. This method executes after parsing, but before routing or translation.

Syntax **Method Validate as Integer**
 [statements]
End Method

Valid Values *statements* any valid Edibasic statements, but typically includes a call to the **Exception** function

Notes You attach validation methods to input definitions, allowing you to validate incoming data at all levels of wrappers and documents, including loops. When there is an exception, you can use the **Exception (on page 734)** function within this method to determine whether to accept the document with errors, reject the document, or abort the interchange. When a composite element or simple element has methods created for both composite/element validation and for segment/composite detail validation, the segment/composite detail method takes precedence. To query values on an element or composite definition, you must use the **Element\$** function, because element and composite validation occurs during parsing before the segment is stored in the DES.

You can also use validation methods in a map. These would execute after the parsing phase.

You can invoke a document-level reject only when you use this method during parsing, not mapping.

See Also **Method ValidateAll as Integer** (on page 717)

Example

The following example checks the value of the current element against incoming values in a cross-reference table using the function, **XRef\$** (on page 759), in order to validate incoming ID codes. Also note that you should declare and initialize the integer variable, **IntVar**.

```
Method Validate as Integer
    If XRef$('UOM',Element$) = '' then
        IntVar = Exception (...)
    End If
```

End Method

Method ValidateAll as Integer

Description	Allows users access to Edibasic during parsing, in order to create custom user-validation of the incoming wrapper and/or document data. It is used for repeating entities and executes after all occurrences have been parsed and validated. This method executes after parsing, but before routing or translation.
Syntax	Method ValidateAll as Integer [<i>statements</i>] End Method
Valid Values	<i>statements</i> any valid Edibasic statements, but typically includes a call to the Exception function
Notes	You attach this validation method to input definitions of repeating entities. It provides an opportunity for final validation after all occurrences have been parsed and validated. If there is an exception, you can use the Exception (on page 734) function within this method to determine whether to accept the document with errors, reject the document, or abort the interchange. You can also use validation methods in a map. These would execute after the parsing phase. You can invoke a document-level reject only when you use this method during parsing, not mapping.
See Also	Method Validate as Integer (on page 716)

Example

The following example checks all occurrences of RMR.4 (monetary amount) in the RMR loop after all occurrences have been parsed. You would create this method for the RMR loop, not the RMR segment, which doesn't repeat. Since RMR.4 is optional, and you must have the data, you are checking to make sure that data for this element exists before you translate. Also note that you should declare and initialize the integer variable, IntVar.

```
Method ValidateAll as Integer
    If RMR.4 = '' then
        IntVar = Exception (...)
    End If
End Method
```

Statement Syntax

The following statements allow you to specify certain actions. They are presented in alphabetical order.

Const statement

Description	Provides a named alias for a constant integer or string value.
Syntax	Const <i>const-name</i> = <i>expression</i> [, <i>const-name</i> = <i>expression</i>]...
Valid Values	<i>const-name</i> any alias name that conforms to variable naming requirements. <i>expression</i> integer or string constants.
Notes	Subsequent uses of the constant name result in a direct substitution of the value. Const statements may be declared once for an object, such as, document, segment, or element, and then used by all methods within that object and any of its subordinated objects. Once you use a constant name, you may not reuse it as a variable name.

Example

The following example shows how you might establish constants for use throughout your mapping instructions.

```
Const MAXCOUNT = 64
Const LABEL = "PO #:"
```

Dim statement

Description	Declares named local variables.
Syntax	Dim <i>var-name</i> [As <i>type</i>] [, <i>var-name</i> [As <i>type</i>]] . . .
Valid Values	<i>var-name</i> any valid variable. <i>type</i> any valid data type.

Notes Variable names are NOT case-sensitive.

If you do not declare a specific data type, the default type of Variant is used. You should declare all variables other than variables containing dates or times with **Dim** or **Global** statements, although Edibasic does not require that you do so. (Variables with date or time values should be initialized using the **Now** function or the **Value** function with an appropriate format.) The scope of variables declared with the **Dim** statement, depends on the object for which they are declared, such as, the document, the loop/group, the segment, or the element. Once declared for an object, the variable is accessible for the scope of that object and any subordinate objects. For example, by adding a **Dim** statement to the **Variables** tab of a loop, that variable is created once and is available during generation of multiple repetitions of that loop. If the same variable name is redeclared for a subordinate object, then a new variable is created with the same name that will hide the original variable within the scope of the subordinate object.

See Also **Global** (on page 721), **Variable Names** (on page 696), **Variable Data Types** (on page 696)

Example

The following example declares a variable A and a variable B as different types.

```
Dim A as String
Dim B as Integer
```

Do-Loop statement

Description Causes a group of statements to be executed repeatedly. Although you do not have to use a condition with **While** or **Until**, you may control execution of the statement better by evaluating a condition before (syntax type 1) or after (syntax type 2) the statements are executed.

Syntax

type1	Do [{ While Until } <i>condition</i>] [<i>statements</i>] Loop
type2	Do [<i>statements</i>] Loop [{ While Until } <i>condition</i>]

Valid Values

<i>condition</i>	anything that can be evaluated as true or false, including expressions with or without relational operators and conditions with or without logical operators.
<i>statements</i>	any valid Edibasic statements.

Notes	When While is used, the loop is executed as long as the conditions evaluates to true, stopping only when the condition evaluates to false. When Until is used, the loop is executed as long as the condition evaluates to false, stopping only when the condition evaluates to true.
Warning	You must make sure that either the condition will eventually cause the loop to stop executing or that you use an Exit Do statement to force it to do so. You must use the Exit Do statement if you do not use While or Until . Otherwise, the loop will execute forever.
See Also	While statement (on page 728), Exit statement (on page 729)

Example

The following example calls the **InitOcc** (on page 741) function, uses a **Do-While** statement to print the first element in the N1 segment, and after all occurrences have been printed, calls the **ResetOcc** (on page 749) function.

```
Rslt = InitOcc(N1)
Do While NextOcc(N1)
    Print N1.1
Loop
Rslt = ResetOcc(Doc)
```

For Each statement

Description	Causes repeated execution of a group of statements based on the number of repeating composite or simple elements, segments or loops in the input.	
Syntax	type1	For Each <i>name</i> Segment [<i>statements</i>] Next Segment
	type2	For Each <i>name</i> Loop [<i>statements</i>] Next Loop
	type3	For Each <i>name</i> Element [<i>statements</i>] Next Element
Valid Values	<i>name</i>	any valid segment or loop in the DOC , WRAPPER , SSEDOC or SSEWRAP in the data element store

Notes The **For Each** statement executes for each occurrence of the composite or simple element, segment or loop. **Next** ends the repetition and passes control to the **For Each** statement to check for another occurrence. When there are no more occurrences, **For Each** passes control to the statement immediately following **Next**. You may use the **Exit For** statement as an alternative way to exit the **For Each** statement. The **Exit For** passes control to the statement immediately following **Next**.

See Also **For statement** (on page 730); **Exit For** (on page 729)

Example

The following example counts the number of N1 loops.

```
Dim Cnt as Integer
Cnt = 0
For Each N1 Loop
    Cnt = Cnt + 1
Next Loop
```

The previous example has the same result as the **Count** function, used as follows:

```
Dim Cnt as Integer
Cnt = Count (N1)
```

Global statement

Description	Declares named global variables within a map or user validation routine that can be accessed within subsequently processed maps or validation methods, all of which must be in the same input stream.
Syntax	Global <i>var-name</i> [As <i>type</i>] [, <i>var-name</i> [As <i>type</i>]]...
Valid Values	<i>var-name</i> any valid variable. <i>type</i> any valid data type.

Notes You can use global variables to pass information from one map to another, such as, accumulating totals in a document map and using a wrapper map to access the totals and place them in the wrapper trailer, or accumulate hash totals to store in a subsequent proprietary acknowledgment. You manipulate the information in the variables using Edibasic.

Global variables must be declared on the Variables page of the Edibasic Edit window, accessible either from the Map window or from the Wrapper Set, Document, Segment, Composite, and Element Maintenance windows for validation routines. You cannot declare global variables within a method.

Like other variables, global variable names are not case-sensitive. The following declarations refer to the same variable:

```
Global MyVar As String
and
Global myvar As String
```

If you do not declare a specific data type, the default type of Variant is used.

The lifetime of variables declared with the Global statement is from the point of first reference continuing until all processing for that input stream is complete. The scope of a global variable is the same as that of a local variable, those declared with a Dim statement. In order to access a global variable, you must set the scope with a declaration within user validation or EDI mapping for each document or wrapper that contains the Edibasic reference to the variable. To set the scope of a global variable, you declare it on a variables tab using the Global statement for a particular entity, such as a wrapper, document, loop or group, or segment. You can set the scope once at a high level, such as document, or set it multiple times as needed at lower levels, such as different segments.

Warning Although the lifetime of a global variable ensures that it exists for the duration of the processing stream, the scope may prevent you from using the variable. When you attempt to access a variable, it must be visible within the scope you have set. For example, if you declare a global variable for a segment, any Edibasic code accessing the variable should be at the segment or element levels. If the Edibasic code is at the document or loop/group level, the scope will prevent you from accessing that variable. When you try to access a variable, global or local, that is outside the scope of the variable, the TRM will create a temporary variable of that name and attempt to use the temporary variable.

See Also *Dim* (on page 718), *Variable Names* (on page 696), *Variable Data Types* (on page 696)

Example

The following example declares variables within wrapper and document definitions to hold values during parsing. Mapping instructions within the wrapper and document maps use the values to create output during generation.

On the **Variables** tab of the Edibasic Edit window for the document, you declare:

```
Global MyVariable Bcd
```

On the **Variables** tab of the Edibasic Edit window for the wrapper, you declare:

```
Global MYVARIABLE as BCD
```

You place a value in **MyVariable** with a mapping instruction in the document map. You then access that value with a mapping instruction in the wrapper map.

If statement

Description Causes conditional execution of a group of statements.

Syntax

```
If condition1 Then
    statements1
[Elseif condition2 Then
    statements2]...
[Else condition3 Then
    statements3]
End If
```

Valid Values

condition anything that can be evaluated as true or false, including expressions with or without relational operators and conditions with or without logical operators.

any valid Edibasic statements.

statements

Notes When *condition1* evaluates to true, then *statements1* is executed. Otherwise, each **Elseif** condition (*condition2*) is evaluated if any exists, and if true, then *statements2* is executed. When none of these evaluates to true and an **Else** clause exists, then *statements3* will be executed.

Example

The following example places the value in N1.2 in a variable called SellerName when N1.1 contains "SE" and in a variable called BuyerName when N1.1 contains "BY" and in a variable called Name for all other values.

```
If N1.1 = "SE" Then
    SellerName = N1.2
ElseIf N1.1 = "BY" Then
    BuyerName = N1.2
Else
    Name = N1.2
End If
```

Let statement

Description	Assigns a value or expression to a variable.
Syntax	[Let] <i>var-name</i> = <i>expression</i>
Valid Values	<i>var-name</i> <i>any valid variable.</i> <i>expression</i> <i>any string or numeric expression.</i>
Notes	The keyword Let is optional, and is often not used.

Example

The following example uses two methods to maintain counters, and uses the optional keyword **Let** with the **Count** function.

```
I = I + 1
MapEle = PO1.3
Let CNT = Count (N1)
```

Mid\$ statement

Description	Returns a string contained within another string.
Syntax	<i>string-exp</i> = Mid\$ (<i>variable</i> , <i>n</i> [, <i>m</i>])
Valid Values	<i>string-expr</i> any string expression used to replace characters in the variable. <i>variable</i> any valid Edibasic variable of the type String or Variant of VarType 3 (string) that is to be modified. <i>n</i> beginning byte position for returned value <i>m</i> number of characters to be returned
Notes	The values for <i>m</i> and <i>n</i> must be positive integers. When <i>m</i> is not given, the number of characters replaced defaults to 1.

Example

The following example splits a string into two sub strings where a slash (/) or dash (-) occurs, and performs a cross-reference on the second sub string.

```
Method Start

Rem Starting string is Str, first sub string is str1
Rem Second sub string is str2

Dim Str As String
Dim Str1 As String
Dim Str2 As String
Dim I As Integer
Dim Found As Integer
Dim Chr As String

Str = "FIRST/SECOND"      ' Could also be FIRST-SECOND

Found = 0
For I = 1 To Len(Str)
    Chr = Mid$(Str,I,1)
    If (Chr = '/') or (Chr = '-') Then
        Found = 1
        Exit For
    End If
Next I

If Found = 1 then
    Str1 = Mid$(Str,1,I-1)
    Str2 = Mid$(Str,I+1,Len(Str)-I)
Else
    Str1 = Str
    Str2 = ''
End If

Str2 = Xref$('XREF-TABLE-NAME',Str2)

End Method
```

Print statement

Description	Prints a list of expressions to the translation report.
Syntax	Print [[<i>expression</i>] [{ ; , }]]..
Valid Values	<i>expression</i> any valid expression.

Notes Print statements can follow one another , and the options allow you to control the format as follows:

- ; print immediately next to preceding printed information (no spaces in-between)
- , print in columns of 14 characters each

The default is to print each expression on a new line. If there is no expression, you print a blank line.

Example

The following example shows different types of Print statements.

```
Print "Column1", "Column2"        ' prints in columns
Print "Edi";"kit"                ' no spaces
Print                                ' blank line
```

Rem statement

Description Adds non-executable comments to code

Syntax **Rem**| *remark*

Valid Values *remark* any displayable information

Notes **Rem** is used on a line by itself, whereas the single quote (apostrophe) can be used either on a line by itself or after another statement on the same line.

Example

The following examples show the use of the syntactic options.

```
Print "This is valid"                ' in-line comment
Print "This isn't valid"            Rem error
Rem This is valid
' This also is valid
```

Select Case statement

Description Allows conditional execution of multiple paths of code depending on the evaluation of a controlling expression (expression).

Syntax **Select Case** *expression*
 [**Case** *case-expressions statements*]. . .
 [**Case Else** *statements*]
 End Select

Valid Values	<i>expression</i>	any string or numeric expression.
	<i>case-expressions</i>	any <i>case-expression</i> that can be evaluated as true or false.
	<i>case-expression</i>	one or more of the following forms (multiple forms must be separated by commas):
		▪ <i>expression</i>
		▪ Is relational-operator expression
		any valid Edibasic statements
	<i>statements</i>	

Notes The *case-expressions* in each **Case** block are evaluated in turn until one evaluates to true. If one evaluates to true, the statements for that block are executed. If none of the expressions evaluates to true, the statements in the **Case Else** block are executed, if one exists.

See Also **If statement** (on page 723), **Method Condition As Integer** (on page 712)

Example

The following example uses a case structure to create output only when input elements 7, 11 or 18 the BS1 segment have values. You would typically write this condition in the Edibasic Edit window on the output segment or loop.

```

Condition = 1      'initialize value to more occ.
Select Case Idx
  case 1          'if 1st occ. of N1, do the following
    Select Case Idx2
      case 1      'if 1st occ. of PER
        If BS1.7 = '' then      'if no data
          Condition = 0
        End If
        Exit Method
      case 2      'if 2nd occ. of PER
        If BS1.11 = '' then
          Condition = 0
        End If
        Exit Method
    End Select

  case 2          'if 2nd occ. of N1
    Select Case Idx2
      case 1
        If BS1.18 = '' then
          Condition = 0
        End If
        Exit Method
    End Select
End Select

```

While statement

Description	Causes repeated execution of statements until a condition evaluates to false.	
Syntax	While <i>condition</i> [<i>statements</i>] Wend	
Valid Values	<i>condition</i>	anything that can be evaluated as true or false, including expressions with or without relational operators and conditions with or without logical operators.
	<i>statements</i>	any valid Edibasic statements
Notes	This structure can be nested. It has the same effect as the Do-Loop using While . The Do-Loop forces more structure but also gives you more options and greater flexibility.	
See Also	Do-Loop <i>statement</i> (on page 719)	

Example

The following example calls the **InitOcc** function, uses a **While** statement to print the first and second elements in the N1 segment for all occurrences.

```
Rslt = InitOcc(N1)
While NextOcc(N1)
    Print N1.1; ":"; N1.2
Wend
```

Function Syntax

Function syntax for Edibasic requires that a value always be returned. Sometimes this value is a string (all functions with **\$** return string values) and sometimes this value is an integer.

NOTE: When Edibasic does not use the returned value, as is the case with **SetField\$**, a dummy variable is used to store the unused returned value.

Abs function

Description	Returns the absolute value of a number, which is its value without regard to sign.	
Syntax	Abs (<i>number</i>)	
Valid Values	<i>number</i>	any valid numeric expression
Examples		

The following example finds the absolute value of an integer.

```
Dim A, B
B = -35
A = Abs(B)
Print A           'result is 35
```

Asc function

Description Returns the numeric ASCII code value of the first character in a string.

Syntax **Asc** (*string*)

Valid Values *string* string expression

See Also **Chr\$** (on page 731), **Format\$** (on page 737), **Str\$** (on page 754), **Val** (on page 757)

Examples

The following example returns the ASCII code for the first character to the variable, I.

```
Dim I As Integer
I = Asc('A')           ' I now = 65
```

Avg function

Description Returns an average numeric value of all occurrences of a data-element within the current scope.

Syntax **Avg** (*data-element*)

Valid Values *data-element* valid input data element identified with a minimum of segment ID and element sequence number. For more information about element names, refer to the topic, **Data Element Names** (on page 698).

See Also **Count** (on page 732), **Min** (on page 745), **Max** (on page 744), **Sum** (on page 755)

Examples

The following example returns the average of all occurrences of the element. To store the returned value in an element, you must use the **Str\$** function to convert the numeric value returned by the **Avg** function to a string value, which is required for all elements.

```
MapEle = Str$(Avg(SEG.3))
```

Exit statement

Description Causes immediate termination of the enclosing method, **Do-Loop** statement, or **For** statement.

Syntax	Exit {Do For Method}
Notes	You must use this Exit statement if you need to expressly terminate a Do-Loop statement, a For statement, or a method, when any one of these may not terminate based on other logic.
See Also	Do-Loop <i>statement</i> (on page 719), For <i>statement</i> (on page 730), Method (on page 712)

Example

The following example uses the **Exit** statement to break out of the N1 loop.

```
For Each N1 Loop
    If N1.1 = "SE" Then
        Name = N1.2
        Exit For
    End If
Next Loop
```

For statement

Description	Causes repeated execution of a group of statements for a specified number of times.
Syntax	For <i>var-name</i> = <i>start</i> To <i>end</i> [Step <i>size</i>] [<i>statements</i>] Next <i>var-name</i> [, <i>var-name</i> ...]
Valid Values	<i>var-name</i> any valid variable, used as a loop counter of data type integer or variant. <i>start</i> the initial value of the counter. <i>end</i> the final value of the counter. <i>size</i> the size of the increments added to the counter. If not specified, defaults to 1.
Notes	The For statement executes for each value in size increments, from start until end, including both start and end. For statements can be nested.
See Also	For-each <i>statement</i> (on page 720)
Example	

The following example shows nested **For** statements. Note that if nested statements end at the same place, you can use one **Next** statement and reverse the counters.

```
For I = 1 To 100
  Print I
  For J = 1 To 100
    Print J
  Next J,I
```

Chr\$ function

Description	Converts a numeric ANSI code value to an ASCII character.
Syntax	Chr\$ (<i>value</i>)
Valid Values	<i>value</i> any digit between 0 and 255, inclusive.
See Also	Asc (on page 729), Format\$ (on page 737), Str\$ (on page 754), Val (on page 757)
Examples	

The following example returns the ASCII character (A), equivalent to the code 65.

```
Print Chr$(65)           'prints A
```

Component\$ function

Description	Returns the element value for a specific component of the current composite element.
Syntax	Component\$ (<i>expression</i>)
Valid Values	<i>Expression</i> any valid Edibasic expression that returns a value equivalent to the sequential order (based on a 1-relative index) that identifies the component within the composite element.
Notes	If you call Component\$ from a method other than Validate or if you call it from something other than a composite element, such as a component or simple element, then Component\$ will return a null string.
See Also	Exception (on page 734)
Examples	

The following example validates that if the first component contains a value of '1', the second component is null, which is represented by 2 single quotes (if not, it invokes the exception function). Alternatively, it validates that if the first component contains a value of '2' and the third component is null.

```
Method Validate as Integer
If Component$(1) = '1' and Component$(2) = '' then
    Validate = Exception(...)
Elseif Component$(1) = '2' and Component$(3) = '' then
    Validate = Exception(...)
End If

End Method
```

Convert\$ function

Description	Converts an input string expression using a date or number format to an output date or number format.
Syntax	Convert\$ (<i>expression</i> , <i>format1</i> , <i>format2</i>)
Valid Values	<i>expression</i> string expression <i>format1</i> any date or number format <i>format2</i> any date or number format
Notes	The date or number formats must follow the requirements listed under the function Format\$ and be enclosed in either single or double quotation marks.
See Also	Format\$ (on page 737), Value (on page 757)
Examples	

The following example changes the string "101596" from a mmddy format to a yymmdd format, returning a string of "961015":

```
MapEle = Convert$("101596", 'mmddy', 'yymmdd')
```

Count function

Description	Returns the number of all occurrences of a data-element within the current scope.
Syntax	Count (<i>loop or segment ID</i>)
Valid Values	<i>loop or segment ID</i> valid input loop or segment ID, also called a tag. The loop and segment IDs are part of the document definition. For more information, refer to the topic, Tag (on page 564), in the Document window reference information.

Notes The count function returns an integer value. When you use it within a method that requires a string value, such as **MapEle**, you must first use another function to convert it, **Str\$**.

See Also **Avg** (on page 729), **Min** (on page 745), **Max** (on page 744), **Sum** (on page 755)

Examples

The following example returns the number of occurrences of the IT1 loop. You must also use the **Str\$** function to change the integer value of IT1 to a string value, which is required by **Method MapEle As String**.

```
MapEle = Str$(Count(IT1))
```

DateSerial function

Description Creates a date variant from year, month, and day integer values.

Syntax **DateSerial** (*year,month,day*)

Valid Values

<i>year</i>	any integer expression producing a 4-digit year
<i>month</i>	any integer expression producing a value 1 - 31
<i>day</i>	any integer expression producing a value 1 - 12

Notes When you use integer values that produce an invalid date, such as February 31 (2004,2,31), they are adjusted forward. February 31, 2004 would be created as a date variant of March 2, 2004, allowing for the leap year date of February 29.
IMPORTANT: There may be a platform-dependent restriction for this function on the earliest supported date. On Windows, it may only work for a relatively small range of dates starting Jan 1 1970. On other platforms, it works for dates starting Jan 1 1900.

See Also **Day** (on page 733), **Month** (on page 746), **Value** (on page 757), **Now** (on page 748), **Weekday** (on page 759), **Year** (on page 760)

Examples

The following example returns a date variant composed of integer values extracted from the system datetime using the Year, Month and Day functions with the Now function.

```
Dim NowDate
```

```
NowDate = DateSerial(Year(Now), Month(Now), Day(Now))
```

Day function

Description Returns the day of the month as an integer value (1-31) from a date variant.

Syntax **Day** (*date variant*)

Valid Values *date variant* Value created by **DateSerial**, **Now** and **Value** functions

Notes The functions **DateSerial**, **Now** and **Value** return date variant values, which can be used to produce values for this function. When you use it within a method that requires a string value, such as **MapEle**, you must use another function, such as **Str\$**, to convert the integer value to a string first.

See Also **DateSerial** (on page 733), **Month** (on page 746), **Value** (on page 757), **Now** (on page 748), **Weekday** (on page 759), **Year** (on page 760)

Examples

The following example returns the number of the day from the system datetime using the **Now** function.

```
Dim I as Integer
```

```
I = Day(Now)
```

Element\$ function

Description Returns the current element value when called with the **Validate Method** during parsing.

Syntax **Element\$**

Notes If you call **Element\$** from a method other than **Validate**, or if you call it from something other than a simple element, such as a composite, then **Element\$** will return a null string.

See Also **Exception** (on page 734)

Examples

The following example checks the value of the current element against incoming values in a cross-reference table in order to validate incoming ID codes:

```
Method Validate as Integer
    If XRef$('UOM',Element$) = '' then
        Validate = Exception (...)
    End If
End Method
```

Exception function

Description Invokes an action for a specific exception (error) number.

Syntax **Exception** (*exception-no*, *action*, *text*)

Valid Values

<i>exception-no</i>	0 through 30000, inclusive.
<i>action</i>	A_ACCEPT , accept with errors
	A_REJECT , reject document
	A_ABORT , report translation failure, abort processing

text any text to be printed on the exception report

Notes Exceptions in the range of 20001 to 30000 inclusive are user-defined. If the action is zero, then the default action is used. User-defined exceptions default to an action of **A_REJECT**. Undefined exceptions default to an action of **A_ABORT**. In all cases, the included text is printed on the exception report together with the exception number and exception description (default and user-defined exceptions).

Examples

The following example invokes an abort action for a user-defined map error that will print on the translation report:

```
I = Exception (20001, A_ABORT, "Fatal Map Error")
```

Field\$ function

Description Returns a string value for the built-in field indicated by the field-id. These field values are parsed from the input data, taken from configurations or created by the TRM for status and statistics.

Syntax **Field\$** (*field-id*)

Valid Values Integer Constant
for *field-id* (use an integer or a constant)

1	F_SEND_ID
2	F_SEND_QUAL
3	F_SEND_INTID (formerly F_SEND_SUBID)
4	F_REC_ID
5	F_REC_QUAL
6	F_REC_INTID (formerly F_REC_SUBID)
7	F_CONTROL_REF
8	F_PASSWORD
9	F_PASSWORD_QUAL
10	F_ICH_VERSION

11	F_COUNT
12	F_DATE
13	F_TIME
14	F_ACK_REQUEST
15	F_TEST_IND
16	F_SET_ID
17	F_DOC_VERSION
18	F_RELEASE
19	F_AGENCY
20	F_ASSOC
21	F_FUNC_GRP_ID
22	F_RECORD_TAG
23	F_TAG_DELIM
24	F_ELE_DELIM
25	F_COMP_DELIM
26	F_SEG_TTRM
27	F_DECIMAL_MARK
28	F_RELEASE_CHAR
29	F_SEND_INTID2
30	F_REC_INTID2
31	F_SEND_MAILBOX
32	F_REC_MAILBOX
33	F_ACK_EXPECTED
34	F_REPEAT_SEPARATOR
101	F_ACK_COUNT
102	F_ACK_STATUS
103	F_ACK_SUBJECT_DOCID
104	F_USER_FIELD1
105	F_USER_FIELD2
106	F_USER_FIELD3
107	F_USER_FIELD4
108	F_VAL_FIELD1
109	F_VAL_FIELD2
110	F_VAL_FIELD3
111	F_VAL_FIELD4
112	F_ACK_CONTROL_REF

Notes You can use either the integer value or the built-in constants in your mapping code.

See Also **Str\$** (on page 754), **SetField** (on page 751)

Examples

The following example moves the value in **F_COUNT** to the current output element.

```
MapEle = Field$(F_COUNT)
```

Format\$ function

Description Returns a formatted string of type datetime or number using format. The purpose of this function is to format non-string data types. Values are determined to be numeric if the format contains a 0, #, or *. Otherwise, values are considered datetime.

NOTE: Format\$ affects only the display format. It does not change the underlying positive or negative value of a number.

Syntax `Format$ ({datetime| number}, format)`

Valid Values

<i>datetime</i>	Date or time data (variant values of VTDate type).
<i>number</i>	Any non-string numeric data (format must contain a 0, #, or *) (numeric values of BCD, integer, or string type).
<i>format</i>	Format value (listed below) or variable containing a format value.

Valid Values for Numbers (case-sensitive)

Value	Description
0	Displays a digit or a zero. If the number has fewer digits on either side of the decimal mark than there are zeros in the format, then leading and/or trailing zeros will be displayed. Extra digits to the right will be rounded. Extra digits to the left will be displayed as is.
#	Displays a digit or nothing.
*	Shorthand for a sequence of # characters. The TRM expands this to the number of characters necessary to allow maximum length (18) of the integer or decimal part of the number. For instance *0.00* is equivalent to #####0.00#####. This character should not occur more than once in the integer part or more than once in the decimal part of the format.
.	The decimal will be displayed as the internal field value from the second page of the wrapper definition or from a user exit, with default of period (.).
..	The decimal will be displayed as a period (.).

Valid Values**for Numbers****(case-sensitive)**

Value	Description
.,	The decimal will be displayed as a comma (,).
.v	Implied decimal. Decimal mark will not be displayed.
,	Triad separator. If a digit has been displayed to the left of this character, then display this character, otherwise, don't. If immediately followed by a period (.), then the triad separator will be displayed as a period.
-()	Display character if the number value is negative. Default for negative numbers is minus sign, -.
+	Display character if the number value is positive or zero. The default for positive numbers is no sign.
+-	Always display a sign. Place before or after format to indicate leading or trailing location of sign.

Valid Values for**Date/Time****(case-sensitive)**

Format	Description
d	Display the day without leading zeros (1-31).
dd	Display the day with leading zeros (01-31).
m	Display the month as a number without leading zeros (1-12).
mm	Display the month as a number with leading zeros (01-12).
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as the entire name (January - December).
yy	Display the year as two digits (00-99).
yyyy	Display the year as four digits.
ccyy	Display the century and year as four digits.
h	Display the hour without leading zeros (0-23).
hh	Display the hour with leading zeros (00-23).
n	Display the minute without leading zeros (0-59).

**Valid Values for
Date/Time
(case-sensitive)**

Format	Description
nn	Display the minute with leading zeros (00-59).
s	Display the second without leading zeros (0-59).
ss	Display the second with leading zeros (00-59).
AM/PM	Display the hour as a 12-hour clock, displaying 'AM' for times before noon and 'PM' for noon and after.
am/pm	Display the hour as a 12-hour clock, displaying 'am' for times before noon and 'pm' for noon and after.

Notes Input should be a non-string data type or a numeric constant, BCD or integer, although string data types also work if the value is numeric. Datetime values must be a variant type. Variant type variables containing datetime information must be initialized using the Now function or the Value function with a format value. The format values are case sensitive, and with 2 exceptions (AM and PM), all are lowercase. The TRM assumes any other alpha characters are constants and prints them as part of the value.

See Also **Convert\$** (on page 732), **Dim** (on page 718), **Now** (on page 748), **Value** (on page 757)

Examples

The following table shows what strings are returned for the numbers with the associated formats (the default notation for negative numbers is the minus sign, so a minus sign is not required in the format notation):

Values/ Formats	5	-5	.5	1,000
0	5	-5	1	1000
0.00	5.00	-5.00	0.50	1000.00
0.v00	500	-500	050	100000
(\$0.00)	\$5.00	(\$5.00)	\$0.50	\$1000.00
###, .###.,00	5,00	-5,00	,50	1.000,00

The following table shows what strings are returned for the dates or times with the associated formats:

Values/ Formats	030195	090805

mm/dd/yy	03/01/95
m/d/yyyy	3/1/1995
mmm d, yyyy	Mar 1, 1995
yymmdd	950301
h:n:s	9:8:5
hh:nn:ss	09:08:05
hh:nnam/pm	09:08am

The following example converts an integer value, created by the **Min** function, to a formatted string, which is required for all output elements. The **Min** function determines the minimum value for all occurrences of an element within repeating entities for the current scope.

```
MapEle = Format$(Min(PO1.4), '0000000000000000.v00')
```

This code determines the least of all unit cost values for these occurrences of the PO1.4 element and reformats it to an implicit decimal before storing it as a string in the output element.

The following example converts a numeric value, stored as a string in the PO1.4 element, to a formatted string:

```
MapEle = Format$(PO1.4, '00000000.v00')
```

Hour function

Description	Returns the hour as an integer value (0-23) from a date variant.
Syntax	Hour (<i>date variant</i>)
Valid Values	<i>date variant</i> value returned by TimeSerial , Now and Value functions.
Notes	The functions TimeSerial , Now and Value return date variant values, which can be used to produce values for this function. When you use it within a method that requires a string value, such as MapEle , you must first use another function, such as Str\$, to convert the integer value to a string.
See Also	Minute (on page 746), Now (on page 748), Second (on page 751), TimeSerial (on page 755), Value (on page 757)

Examples

The following example returns the hour of the day from the system datetime using the **Now** function.

```
Dim I as Integer
```

```
I = Hour(Now)
```

InitOcc function

Description	Initializes the occurrences of repeating entities in the Data Element Store. This prepares for subsequent repetitions of the given loop or segment by returning the nested repetition level of the loop or segment.
Syntax	InitOcc (seg)
Valid Values	seg any valid segment ID in the data element store.
Notes	The initialization process 1) sets the pointer level of the repeating segment or loop in which the segment occurs, and 2) sets the repetition of the segment or loop to zero. You usually follow this function with the NextOcc function, which returns the first occurrence of the segment. For more information, refer to the topic, Advanced Mapping Techniques (on page 185).
See Also	NextOcc (on page 747), Occurrence (on page 748), ResetOcc (on page 749)

Examples

The following example sets the occurrence pointer of the N1 loop to zero. Notice that the area number and loop sequence number are provided, so that you are assured of accessing the desired N1 loop.

```
Rslt = InitOcc(N1{1,90})
```

InStr function

Description	Returns the beginning position of a string, S2, within another string, S1, with the option to specify whether the search should be case sensitive, which is the default.
Syntax	InStr (s1, s2 [,c])
Valid Values	s1 string expression. s2 string expression for which you are searching contained in string s1. c optional integer expression to specify case sensitivity. When the integer is missing or equals zero, the search will use upper and lower case values. When the integer is 1, the search will not use upper and lower case values.
Notes	The InStr function returns an integer value representing the first byte of the first matching string. Note that all string values must be in quotes.

Examples

The following example returns the position of the first 'i' that it encounters within a string, without regard to case. It should return the position of the pronoun I, which is 1.

```
Dim I as Integer
```

```
I = InStr("I am in this string","i", 1)
```

The next example returns the position of the first "i" that it encounters within a string, using the lower case value. It should return the position of the "i" in the word "in", which is 6.

```
I = InStr("I am in this string","i")
```

IsNull function

Description	Determines whether an element or a variable contains no data.	
Syntax	IsNull ({ <i>element</i> <i>variable</i> })	
Valid Values	<i>element</i>	valid input data element identified with a minimum of segment ID and element sequence number. For more information about element names, refer to the topic, Data Element Names (on page 698).
	<i>variable</i>	named storage location
Notes	Returns non-zero if the element or variable contains a null value, otherwise returns zero. If the variable is a string, then zero length is considered null. Other data types do not have null values.	
See Also	IsNumeric (on page 742)	
Examples		

The following example checks to see if the element N1.2 has any data associated with it, and if not (if it is null), then the program will execute the statements (not given) following the **If...Then** clause.

```
If IsNull(N1.2) Then
```

IsNumeric function

Description	Indicates whether an element or a variable can be converted to a numeric data type. Returns a non-zero value if the variable is numeric or can be converted to a numeric type.	
Syntax	IsNumeric ({ <i>element</i> <i>variable</i> })	
Valid Values	<i>element</i>	valid input data element identified with a minimum of segment ID and element sequence number. For more information about element names, refer to the topic, Data Element Names (on page 698)
	<i>variable</i>	named storage location

Notes Integer and BCD types are always numeric. String types may be converted to numeric if the contents consist of a valid number constant with optional white space before and after the number. Variant types are numeric if the stored value is numeric.

See Also **IsNull** (on page 742)

Examples

The following example uses **IsNumeric** to determine if the value in the variable A can be converted to an integer. If it can, the **Val** function is used to convert the value to an integer that is stored in variable B.

```
Dim A As String, B As Integer

A = " 436"

If IsNumeric (A) Then

    B = Val (A)

End If
```

LCase\$ function

Description Returns a string from string whose characters are all converted to lowercase.

Syntax **LCase\$** (*string*)

Valid Values *string* string expression.

See Also **UCase\$** (on page 756)

Example

The following example converts the characters in the element 01.2 to all lowercase. Notice that the segment ID, 01, is enclosed in brackets because it does not begin with an alpha character. For more information about the syntax of element names, refer to the topic, **Data Element Names** (on page 698).

```
MapEle = LCase$ ([01].2)
```

Left\$ function

Description Returns a string containing the leftmost *n* characters of string.

Syntax **Left\$** (*string*,*n*)

Valid Values *string* any string expression.
n integer indicating how many characters to return, between 0 and 65,535, inclusive.

Notes If *n* is 0, the returned value is a zero-length string.

See Also [Len](#) (on page 744), [Mid\\$](#) (on page 745), [Right\\$](#) (on page 750)

Example

The following example returns the leftmost 4 characters of the string value, "19950103".

```
MapEle = Left$ ("19950103",4) 'returns 1995
```

Len function

Description Returns the number of characters in a string.

Syntax **Len** (*string*)

Valid Values *string* string expression.

[Left\\$](#) (on page 743), [Mid\\$](#) (on page 745), [Right\\$](#) (on page 750)

See Also

Example

The following example returns the number of characters in the string, test string.

```
Print Len("Test String") 'prints 11
```

LTrim\$ function

Description Returns a string with leading spaces of string removed

Syntax **LTrim\$** (*string*)

Valid Values *string* string expression

See Also [RTrim\\$](#) (on page 751), [Trim\\$](#) (on page 756)

Example

The following example removes leading spaces from the string, " Data".

```
MapEle = LTrim$ (" Data") 'returns DATA
```

Max function

Description Returns the maximum value of all occurrences of data-element within the current scope.

Syntax **Max** (*data-element*)

Valid Values *data-element* valid input data element identified with a minimum of segment ID and element sequence number. For more information about element names, refer to the topic, **Data Element Names** (on

page 698).

See Also **Avg** (on page 729), **Count** (on page 732), **Min** (on page 745), **Sum** (on page 755)

Example

The following example places the highest value of all occurrences of the element PO1.3 in the variable M.

```
Dim M As Integer
```

```
M = Max(PO1.3)
```

Mid\$ function

Description Returns a string containing *m* characters from *string*, starting with the *n*th character.

Syntax **Mid\$** (*string*, *n* [,*m*])

Valid Values *string* string expression.

n integer indicating the character position to start within the string.

m integer indicating how many characters to return.

Notes Both *n* and *m* must be between 0 and 65535, inclusive. When *m* is omitted, then all characters starting with the *n*th character are returned. When *n* is greater than the length of the string, the function returns a zero-length string.

See Also **Len** (on page 744), **Left\$** (on page 743), **Right\$** (on page 750)

Example

The following example extracts the 5th and 6th characters from the string, "19950103", and places it in the current element.

```
MapEle = Mid$("19950103",5,2) 'returns 01
```

Min function

Description Returns the minimum value of all occurrences of data-element within the current scope.

Syntax **Min** (*data-element*)

Valid Values *data-element* valid input data element identified with a minimum of segment ID and element sequence number. For more information about element names, refer to the topic, **Data Element Names** (on page 698).

See Also **Avg** (on page 729), **Count** (on page 732), **Max** (on page 744), **Sum** (on page 755)

Example

The following example places the lowest value of all occurrences of the LIN.3 element in the Variable M.

```
Dim M As Integer
```

```
M = Min(LIN.3)
```

Minute function

Description	Returns the minutes of the hour as an integer value (0-59) from a date variant.
Syntax	Minute (<i>date variant</i>)
Valid Values	<i>date variant</i> value returned by TimeSerial , Now and Value functions.
Notes	The functions TimeSerial , Now and Value return date variant values, which can be used to produce values for this function. When you use it within a method that requires a string value, such as MapEle , you must first use another function, such as Str\$, to convert the integer value.
See Also	Hour (on page 740), Now (on page 748), Second (on page 751), TimeSerial (on page 755), Value (on page 757)

Examples

The following example returns the minutes of the hour from the system datetime using the Now function.

```
Dim I as Integer
```

```
I = Minute(Now)
```

Month function

Description	Returns the month as an integer value (1-12) from a date variant.
Syntax	Month (<i>date variant</i>)
Valid Values	<i>date variant</i> value returned by TimeSerial , Now and Value functions.
Notes	The functions DateSerial , Now and Value return date variant values, which can be used to produce values for this function. When you use it within a method that requires a string value, such as MapEle , you must first use another function, such as Str\$, to convert the integer value to a string.
See Also	DateSerial (on page 733), Day (on page 733), Now (on page 748), Value (on page 757), Weekday (on page 759), Year (on page 760)

Examples

The following example returns the number of the month from the system datetime using the **Now** function.

```
Dim I as Integer
```

```
I = Month(Now)
```

NextOcc function

Description Sets a data element store's occurrences to the next occurrence of the segment for the current repetition level.

Syntax **NextOcc** (*seg*)

Valid Values *seg* any valid segment ID in the data element store.

Notes If another occurrence of the segment exists, it will return 1, otherwise it will return 0. You use this function to position current pointers so you can then query the value of a particular element in the next occurrence of the segment, without generating the segment. For more information about how to use this function, refer to the topic, ***Advanced Mapping Techniques*** (on page 185).

See Also **InitOcc** (on page 741), **Occurrence** (on page 748), **ResetOcc** (on page 749)

Example

The following example uses the variable N1Rslt to hold a value of 0 (no more occurrences) or 1 (at least 1 more occurrence) that can be tested as a condition with the While statement, which will execute another time if the condition is true (evaluates to 1) or not if the condition (evaluates to 0).

```
Dim N1Rslt As Integer      'creates variable N1RSLT

N1Rslt = InitOcc(N1)      'resets occurrence of N1 to zero

N1Rslt = NextOcc(N1)      'sets occurrence of N1 to one,
                          'returns 0 if no data, or 1

While N1Rslt              'executes if 1, not if 0
    ...                    'statements not included
    N1Rslt = NextOcc(N1)   'checks for more data
WEnd

N1Rslt = ResetOcc(Doc)    'resets level to value
                          'prior to call to
                          'InitOcc to process
                          'higher levels
```

Now function

Description	Returns the current system date and time as a date variant, type 4 (VTDATE).
Syntax	Now
See Also	Day (on page 733), DateSerial (on page 733), Hour (on page 740), Minute (on page 746), Month (on page 746), Second (on page 751), TimeSerial (on page 755), Value (on page 757), Weekday (on page 759), Year (on page 760)

Example

The following example uses the Now function to retrieve the current system date and time, and then the Format\$ function to put a string value in the element with the format of mm/dd/yyyy.

```
MapEle = Format$(Now, "mm/dd/yyyy")
```

Occurrence function

Description	Based on the input data, this function returns an integer representing the current occurrence of the specified area of the Data Element Store (DES), with the option to move to a different level within the DOC , WRAP , SSEDOC or SSEWRAP structures.
-------------	---

Syntax	Occurrence (<i>des</i> , <i>level</i>)	
Valid Values	<i>des</i>	area of the Data Element Store <ul style="list-style-type: none"> ▪ WRAP ▪ SSEWRAP ▪ SSEDOC ▪ DOC
	<i>level</i>	integer value that returns occurrences as follows: <ul style="list-style-type: none"> ▪ missing or 0, returns the innermost occurrence ▪ >0, relative to the outermost level ▪ <0, relative to the innermost level
Notes	Use this function as a debug tool to determine the current occurrence of a repeating and possibly nested structure in the DES. The occurrences at various levels create the context list that the TRM uses to know where it is in the nested structures, information, which is also available on the translation report when you print the DES. By default, this function returns the current occurrence at the current level of the input. To move the pointer up or down levels in a nested structure, use the level option. To move lower in the structure, use an integer value that is positive. For example, use the value 3 to move from the outermost level, which is 1, to level 3 within the nested structure. To move up toward the top of the structure from the current level, use a negative integer. For example, use the value -2 to move from the lowest, innermost level, which let us say is 3, to level 1.	
See Also	InitOcc (on page 741), NextOcc (on page 747), ResetOcc (on page 749)	

Examples

The following example returns the occurrence of the current pointer:

```
Dim L as integer
```

```
L = Occurrence(DOC, 0)
```

```
Print "Current occurrence in DOC = ",L
```

ResetOcc function

Description	Resets a data element store's occurrences to the state they were in before the last call to InitOcc .	
Syntax	ResetOcc (<i>des</i>)	
Valid Values	<i>des</i>	area of the Data Element Store (DES): <ul style="list-style-type: none"> ▪ WRAP ▪ SSEWRAP ▪ SSEDOC ▪ DOC

Notes Data element stores for the input wrapper (**WRAP**) and its associated status, statistics and errors (**SSEWRAP**) as well as for the input document (**DOC**) and its associated status, statistics, and errors (**SSEDOC**) are available through Edibasic.

See Also [InitOcc \(on page 741\)](#), [NextOcc \(on page 747\)](#), [Occurrence \(on page 748\)](#)

Example

The following example uses the variable N1Rslt to hold a value of 0 (no more occurrences) or 1 (at least 1 more occurrence) that can be tested as a condition with the While statement, which will execute another time if the condition is true (evaluates to 1) or not if the condition (evaluates to 0).

```
Dim N1Rslt As Integer      'creates variable N1RSLT

N1Rslt = InitOcc(N1)      'resets occurrence of N1 to 0

N1Rslt = NextOcc(N1)      'sets occurrence of N1 to one,
                          'returns 0 if no data, or 1

While N1Rslt              'executes if 1, not if 0
    ...                    'statements not included

    N1Rslt = NextOcc(N1)  'checks for more data

WEnd                       'end of While loop

N1Rslt = ResetOcc(Doc)    'resets level to value
                          'prior to call to
                          'InitOcc to process
                          'higher levels
```

Right\$ function

Description Returns a string that contains the rightmost *n* characters of the value in the string.

Syntax **Right\$ (string, n)**

Valid Values *string* any string expression.
n integer indicating how many characters to return, between 0 and 65,535, inclusive.

Notes If *n* is 0, the returned value is a zero-length string.

See Also [Len \(on page 744\)](#), [Left\\$ \(on page 743\)](#), [Mid\\$ \(on page 745\)](#)

Example

The following example puts the last 6 characters in the element.

```
MapEle = Right$("19950103",6) 'returns 950103
```

RTrim\$ function

Description Returns a string containing the value in string with right, trailing spaces removed.

Syntax **RTrim\$** (*string*)

Valid Values *string* string expression

LTrim\$ (on page 744), **Trim\$** (on page 756)

See Also

Example

The following example removes trailing spaces from the value, "DATA ", puts it in the element.

```
MapEle = RTrim$("DATA ") 'returns DATA
```

Second function

Description Returns the second of the minute as an integer value (0-59) from a date variant.

Syntax **Second** (*date variant*)

Valid Values *date variant* value returned by **TimeSerial**, **Now** and **Value** functions.

Notes The functions **TimeSerial**, **Now** and **Value** return date variant values, which can be used to produce values for this function. When you use it within a method that requires a string value, such as **MapEle**, you will need to use another function first, such as **Str\$**, to convert the integer value to a string.

See Also **Hour** (on page 740), **Minute** (on page 746), **TimeSerial** (on page 755), **Value** (on page 757)

Examples

The following example returns the second of the minute from the system datetime using the **Now** function.

```
Dim I as Integer
```

```
I = Second(Now)
```

SetField function

Description Set the built-in field indicated by the *field-id* to the string value provided.

Syntax **SetField** (*field-id*,*string*)

Valid Values for <i>field-id</i> (use an integer or a constant)	Integer	Constant
	1	F_SEND_ID
	2	F_SEND_QUAL
	3	F_SEND_INTID (formerly F_SEND_SUBID)
	4	F_REC_ID
	5	F_REC_QUAL
	6	F_REC_INTID (formerly F_REC_SUBID)
	7	F_CONTROL_REF
	8	F_PASSWORD
	9	F_PASSWORD_QUAL
	10	F_ICH_VERSION
	11	F_COUNT
	12	F_DATE
	13	F_TIME
	14	F_ACK_REQUEST
	15	F_TEST_IND
	16	F_SET_ID
	17	F_DOC_VERSION
	18	F_RELEASE
	19	F_AGENCY
	20	F_ASSOC
	21	F_FUNC_GRP_ID
	22	F_RECORD_TAG
	23	F_TAG_DELIM
	24	F_ELE_DELIM
	25	F_COMP_DELIM
	26	F_SEG_TTRM
	27	F_DECIMAL_MARK
	28	F_RELEASE_CHAR
	29	F_SEND_INTID2

30	F_REC_INTID2
31	F_SEND_MAILBOX
32	F_REC_MAILBOX
33	F_ACK_EXPECTED
34	F_REPEAT_SEPARATOR
101	F_ACK_COUNT
102	F_ACK_STATUS
103	F_ACK_SUBJECT_DOCID
104	F_USER_FIELD1
105	F_USER_FIELD2
106	F_USER_FIELD3
107	F_USER_FIELD4
108	F_VAL_FIELD1
109	F_VAL_FIELD2
110	F_VAL_FIELD3
111	F_VAL_FIELD4
112	F_ACK_CONTROL_REF

Notes You can use the integer value or the built-in field constants in your mapping code.

See Also **Field\$** (on page 735)

Example

The following example sets the value of the test indicator field to '1'. Note that the returned integer value that is stored in the temporary variable, *I*, is not used by the TRM, but is required for function syntax.

```
I = SetField(F_TEST_IND, '1')
```

Space\$ function

Description Returns a string containing *n* spaces.

Syntax **Space\$ (*n*)**

Valid Values *n* 0 through 65,535, inclusive

See Also **Str\$ (on page 754)**, **String\$ (on page 754)**

Example

The following example right justifies N1.1 in a 30-byte field by calculating the number of spaces required to fill the element, which is 30 minus the length of the element, and then concatenates the returned spaces with the value in N1.1.

```
MapEle = Space$(30-Len(N1.1)) & N1.1
```

Str\$ function

Description Returns a string representing the value of a numeric expression, *x*.

Syntax **Str\$ (*x*)**

Valid Values *x* any Edibasic expression that returns a numeric expression.

See Also **Format\$ (on page 737)**, **Val (on page 757)**

Example

The following example uses the **Str\$** function to convert a numeric expression returned by the **Count** function, which contains the number of occurrences of the N1.1 element.

```
MapEle = Str$(Count(N1.1))
```

StrComp function

Description Compares one string, *S1*, within another string, *S2*, with the option to specify whether the comparison should be case sensitive, which is the default.

Syntax **StrComp (*s1*, *s2* [,*c*])**

Valid Values

- s1* string expression
- s2* string expression
- c* optional integer value to specify case sensitivity. When the integer is missing or equals zero, the search will be case-sensitive, matching upper and lower case values. When the integer is 1, the search will not be case-sensitive.

Notes The StrComp function returns an integer value as follows:

- Less than zero when *s1* < *s2*
- Zero when *s1* = *s2*
- Greater than zero when *s1* > *s2*

Examples

The following example compares the value in the element BEG.5 that contains the purchase order date from an inbound document with the date for the beginning of the year. We must have a value that is equal to or greater than "040101".

```
Dim I as Integer
```

```
I = StrComp(BEG{1,20}.5,"040101")
```

String\$ function

Description Returns a string value made from multiple occurrences, *n*, of another string, *s*.

Syntax **String\$ (*n*,*s*)**

Valid Values *n* number of occurrences that the string is to be repeated.
 s string that it to be repeated to create the new string.

See Also **Space\$** (on page 753)

Examples

The following example returns 12 zeroes to the current element.

```
MapEle = String$(12,0)
```

Sum function

Description Returns the sum of all occurrences of data-element within the current scope.

Syntax **Sum** (*data-element*)

Valid Values *data-element* any element in the data element store.

See Also **Avg** (on page 729), **Count** (on page 732), **Max** (on page 744), **Min** (on page 745)

Example

The following example stores the sum of all occurrences of N1.1 in the variable S.

```
Dim S As Integer
```

```
S = Sum(N1.1)
```

TimeSerial function

Description Creates a time variant from the hour, minute, and second integer values.

Syntax **TimeSerial** (*hour,minute,second*)

Valid Values *hour* any integer expression producing a value 0 - 23

minute any integer expression producing a value 0 - 59

second any integer expression producing a value 0 - 59

See Also **DateSerial** (on page 733), **Hour** (on page 740), **Minute** (on page 746), **Now** (on page 748), **Second** (on page 751)

Examples

The following example returns a time variant composed of integer values extracted from the system datetime using the **Hour**, **Minute** and **Second** functions with the **Now** function.

```
Dim NowTime
```

```
NowTime = TimeSerial(Hour(Now), Minute(Now), Second(Now))
```

Trim\$ function

Description Returns a string containing the value in string with right and left spaces removed.

Syntax `Trim$ (string)`

Valid Values *string* string expression

See Also **LTrim\$** (on page 744), **RTrim\$** (on page 751)

Example

The following example removes leading and trailing spaces from the value, " DATA ".

```
MapEle = Trim$(" DATA ") 'returns DATA
```

UCase\$ function

Description Returns a string from string whose characters are all converted to uppercase.

Syntax `UCase$ (string)`

Valid Values *string* string expression

See Also **LCase\$**

Example

The following example converts all of the characters in N3.1 to uppercase.

```
MapEle = UCase$(N3.1)
```

User function

Description Invokes a registered user exit.

Syntax `User (name)`

Valid Values *name* name registered in a user module with a call to ERMRegisterUserExit. Refer to the *MW Translator User Exits Programming Manual* for details.

Examples

The following example calls the registered user exit 'CheckPO' after first setting the global variable GblPO. If the returned value is not zero, an exception is generated

```
GblPO = BEG.3
I = User('CheckPO')
  If I <> 0 then
    I = Exception(...)
  End If
```

Val function

Description Returns the numeric value of a string.

Syntax **Val** (*string*)

Valid Values *string* string expression

See Also **Format\$** (on page 737), **Str\$** (on page 754)

Example

The following example places the numeric value of the element BPR.3 in the variable I.

```
Dim I As Integer

I = Val(BPR.3)
```

Value function

Description Returns a numeric value as data type BCD or datetime from string and applies format.

Syntax **Value** (*string, format*)

Valid Values *string* string expression
format valid format instruction, as described in the topic, **Format\$ function** (on page 737)

See Also **Format\$** (on page 737), **Now** (on page 748)

Notes This function returns a BCD or date variant data type depending on the format. When the format uses zero (0), pound (#), or asterisk (*), the value returned is BCD. Anything else is returned as a date variant.

Example

The following example converts the string value in element AK4.9 to a numeric, BCD value and uses the format specified.

```
Dim A As BCD
```

```
A = Value(AK4.9, "###, ##0.00")
```

The following example converts the string value in element AK4.9 to a date variant value and uses the format specified. We declare the variables with a **Dim** statement without a data type, which creates variant type variables by default. You can then use one of the functions that converts a date variant value to an integer indicating a year, month or day, as we do here, where we use the **Year** function to return the year contained in the date.

```
Dim A, vtYear
```

```
A = Value(BEG.5, "yyyymmdd")
```

```
vtYear = Year(A)
```

You could also easily return an integer indicating the month from the system datetime using the **Month** function with the **Now** function.

```
Dim vtMonth
```

```
vtMonth = Month(Now)
```

VarType function

Description	Returns a numeric value indicating the data type of a variable.	
Syntax	VarType (<i>variable</i>)	
Valid Values	<i>variable</i>	named storage location, using the naming conventions described in the topic, Variable Names (on page 696)
Notes	The following values may be returned. The built-in constant value can be used in mapping code.	

Value	Description	Built-in Constants
0	Null	VTNull
1	Integer	VTInteger
2	BCD	VTBcd
3	String	VTString
4	Date	VTDDate

Example

The following example checks for a date type in the variable, D. If it finds a date type, the **Format\$** function converts it to a string with a format of **mmm d,yyyy**.

```
If VarType(D) = VTDate Then
    MapEle = Format$(D, "mmm d, yyyy")
End If
```

Weekday function

Description	Returns the day of the week as an integer value (1-7) from a date variant, where 1 = Sunday, 2 = Monday, 3 = Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday and 7 = Saturday.
Syntax	Weekday (<i>date variant</i>)
Valid Values	<i>date variant</i> value returned by TimeSerial , Now and Value functions.
Notes	The functions DateSerial , Now and Value return date variant values, which can be used to produce values for this function. When you use it within a method that requires a string value, such as MapEle , you must first use another function, such as Str\$, to convert the integer value to a string.
See Also	DateSerial (on page 733), Day (on page 733), Month (on page 746), Now (on page 748), Value (on page 757), Year (on page 760)

Examples

The following example returns the day of the week from the system datetime using the **Now** function.

```
Dim I as Integer
I = Weekday(Now)
```

Xref\$, XrefR\$ functions

Description	Cross-reference (Xref\$) and reverse cross-reference (XrefR\$) functions that return a string using a table lookup of another string. Xref\$ uses the input column to match against the incoming data and returns the value in the output column. XrefR\$ uses the output column to match against the incoming data and returns the value in the input column. This permits you to use a single table to map in both directions.
Syntax	Xref\$ (<i>table-name</i> , <i>string</i> [, <i>default</i>]) XrefR\$ (<i>table-name</i> , <i>string</i> [, <i>default</i>])
Valid Values	<i>table-name</i> should match the name created with Cross-Reference Table Maintenance and be enclosed in quotes. <i>string</i> any expression that can return a valid string. <i>default</i> any string enclosed in quotes.

Notes Cross-reference tables are names and are defined globally for the server. If the named table is not found at run time or the lookup is unsuccessful, then default, if present, is used. If the table is not found or the lookup is unsuccessful and default is not provided, then a null string is returned. The cross-reference tables may be created and maintained by using the Cross-Reference Table Maintenance window, which is accessible from the Workbench menu.

Example

The following example causes a lookup in a cross-reference table, which should match the value in N4.2 and convert it to the output in the cross-reference table, and if it cannot match the value, it inserts the default value, Michigan.

```
MapEle = XRef$("STATE-TABLE",N4.2,"Michigan")
```

Year function

Description Returns the year as an integer value from a date variant.

Syntax **Year** (*date variant*)

Valid Values *date variant* value returned by **TimeSerial**, **Now** and **Value** functions.

Notes The functions **DateSerial**, **Now** and **Value** return date variant values, which can be used to produce values for this function. When you use it within a method that requires a string value, such as **MapEle**, you must first use another function, such as **Str\$**, to convert the integer value to a string.

See Also **DateSerial** (on page 733), **Day** (on page 733), **Month** (on page 746), **Now** (on page 748), **Weekday** (on page 759), **Value** (on page 757)

Examples

The following example returns the year from the system datetime using the **Now** function.

```
Dim I as Integer
```

```
I = Year(Now)
```

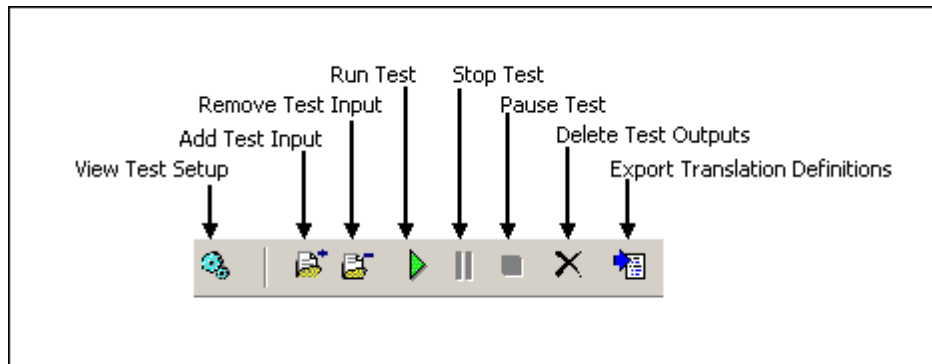
Test Reference

Test Toolbar and Icons

You access the Test window from the menu bar or the toolbar.

Toolbar

The following part of the toolbar appears when you run tests.



Output Icons

The following icons represent the various outputs that the TRM might create during processing. From the Workbench, you can run multiple input files in sequence, which displays multiple sets of output files. When you select an input file after running multiple such tests, the output files related to that input file will be selected and the other output files will be dimmed. The icons vary slightly to indicate the output associated with the selected input versus the dimmed output files, which are associated with other input files.

From Selected Input	From Other Input	Description
		Report file
		Output file
		Backward acknowledgment file


Procedures

You may perform the following task from the Test toolbar.

To Open the Test Window







- From the **View** menu, choose **Test Setup**.

– or –

- From the toolbar, choose the **View Test Setup** button .

Test Menu and Task Icons

There is only one menu specific to the Test window. The following table lists the **Test** menu commands and associated icons.

Command	Shortcut	Icon	Description
Run			Submit the selected input file(s) for translation.
Pause			Pause the translation until the user selects Run or Stop.
Stop			Abort the translation.
Delete Outputs			Delete the output file(s) associated with the selected input file(s).
Add Input...			Add selected input file to the input list.
Remove Input			Remove select input file(s) from the input list.
Force Text			Force the output file(s) to have a new line after each segment for easier viewing.
Auto-delete Outputs			Delete all output from tests when you close the Test window.

Force Text (Test Menu)

This command is on the **Test** menu, and is checked by default. It forces the output to be written with line breaks (carriage return/linefeed) at the end of each segment, which makes the output easier to read when you are viewing the output files during testing.

This overrides the **I/O Mode** if it is set to binary for the identified output wrapper. When the output wrapper has **I/O Mode** set as text, using this flag will have no effect. When the output wrapper has **I/O Mode** set as **Binary**, you will see a difference in the presentation of the output data during testing by checking or clearing this menu item.

Auto-delete Outputs (Test Menu)

This command is on the **Test** menu, and it is not selected by default. When you choose this option, the Workbench will delete all output files, including the processing report, when you rerun the test, close the



window or select **Delete** from the **Test** menu or the **Delete** button from the toolbar. This is to help you keep your system from accumulating test files unnecessarily.

If you want to save a particular file and if **Auto-delete Outputs** is selected, you can double-click to view the file in an editor and use the editor to save it under another name. You should clear the option when you do not want to delete the files.

Procedures

You may perform the following tasks from the **Test** menu.

To Create Output with Line Breaks after Segments

Follow these instructions to force the output to be written with line breaks (carriage return/linefeed) at the end of each segment, which makes the output easier to read when you are viewing the output files during testing. This overrides the **I/O Mode** if it is set to binary for the identified output wrapper.

- From the **Test** menu, select **Force Text**.
A check mark appears next to the command on the menu.

To Automatically Delete Output Files After Testing

Follow these instructions to delete all output files automatically, including the processing report, when

you rerun the test, close the Test window or select **Delete** from the **Test** menu or the **Delete** button from the toolbar.



1 From the **Test** menu, select **Auto-delete Outputs**.



A check mark appears next to the command on the menu.

2 To save a particular file when **Auto-delete Outputs** is selected:

- a) Double-click any of the output files or reports to view the file in an editor.
- b) Save the file under another name.

Test Window

The Workbench uses this information to invoke the Translator Runtime Module (TRM), simulating the process you will use in your production environment, also called runtime processing. The input files display on the input list in the upper pane. Once processing completes, the pertinent information displays on the output list in the lower pane. You can view the input file or any reports or output files created because of processing by double-clicking on the colored text. While a test is in progress, you can choose

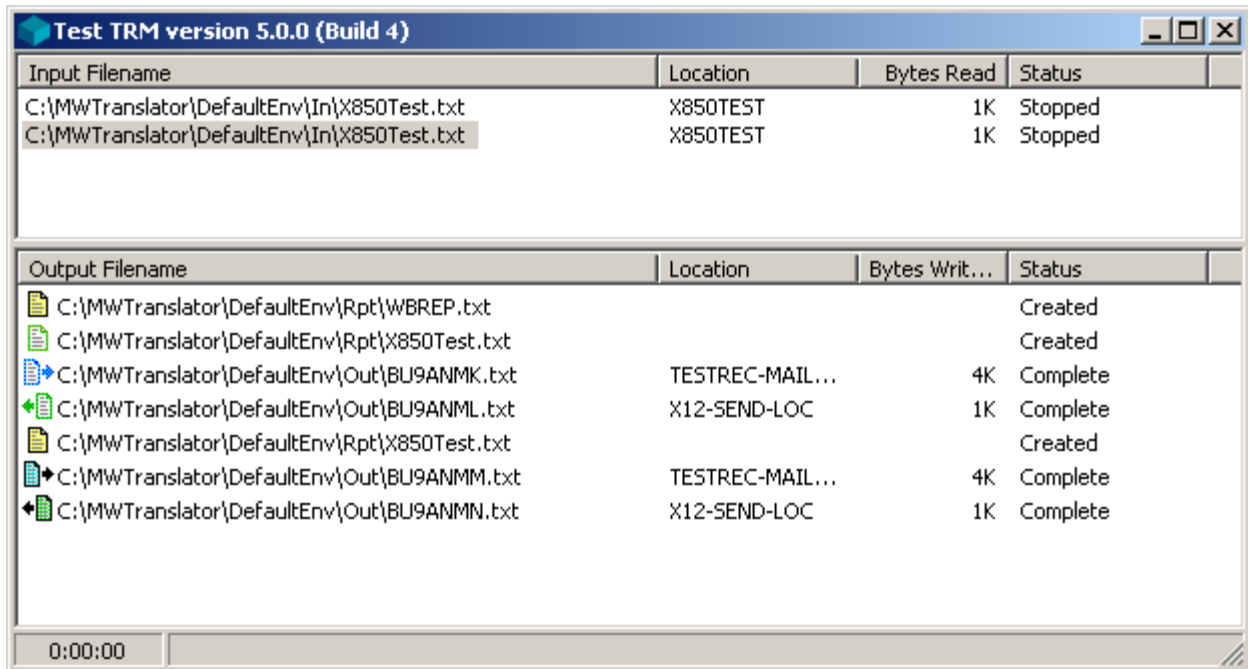
the **Pause Test**  or **Stop Test**  buttons to interrupt or cancel long translations.

IMPORTANT: The **Stop Test** button will work even when endless loops are written in maps or Edibasic, but not when they are written in user exits.

Although you can run only one test at a time, you can queue up several input files, select them, and the Workbench will run them in order. After translation, when you select one of the input files, the Workbench highlights the associated output files. This helps you easily locate the output files, particularly if you run several translations individually and in varying order.

After you have completed a test, you can export all of the definitions that the TRM used to process a test file. Once exported, the definitions can then be imported into a different database. This allows you to easily move definitions from one database to another.

You can access 2 toggle commands from the **Test** menu: **Force Text** and **Auto-delete Outputs**. The **Force Text** command affects the display of the output. The **Auto-delete Outputs** tells the Workbench to delete all the output files created during testing when you close the Test window. When this command is not selected, you must delete the files manually.



The screenshot shows a window titled "Test TRM version 5.0.0 (Build 4)". It contains two tables. The first table, "Input Filename", lists two input files. The second table, "Output Filename", lists seven output files with their locations, bytes written, and status. A timer at the bottom left shows "0:00:00".

Input Filename	Location	Bytes Read	Status
C:\MWTranslator\DefaultEnv\In\X850Test.txt	X850TEST	1K	Stopped
C:\MWTranslator\DefaultEnv\In\X850Test.txt	X850TEST	1K	Stopped


Output Filename	Location	Bytes Writ...	Status
C:\MWTranslator\DefaultEnv\Rpt\WBREP.txt			Created
C:\MWTranslator\DefaultEnv\Rpt\X850Test.txt			Created
C:\MWTranslator\DefaultEnv\Out\BU9ANMK.txt	TESTREC-MAIL...	4K	Complete
C:\MWTranslator\DefaultEnv\Out\BU9ANML.txt	X12-SEND-LOC	1K	Complete
C:\MWTranslator\DefaultEnv\Rpt\X850Test.txt			Created
C:\MWTranslator\DefaultEnv\Out\BU9ANMM.txt	TESTREC-MAIL...	4K	Complete
C:\MWTranslator\DefaultEnv\Out\BU9ANMN.txt	X12-SEND-LOC	1K	Complete

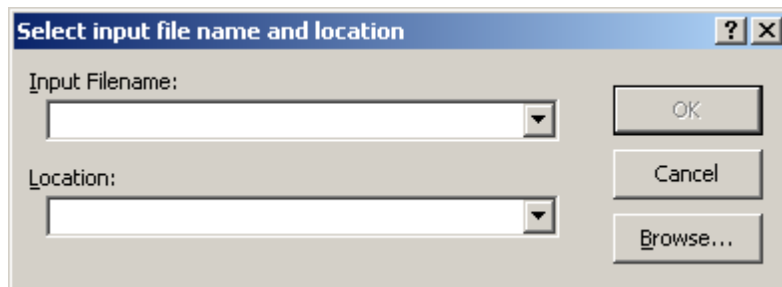
0:00:00

Test Window

Input List (Test)

This list contains a list of the files you have chosen to use as input data during testing. Although you can run only one test at a time, you can queue up several input files, select them, and the Workbench will run them in order.


To add names to the list, select the **Add Test Input** button  from the toolbar. The **Select input file name and location** dialog box appears to allow you to locate an input file from your directories.



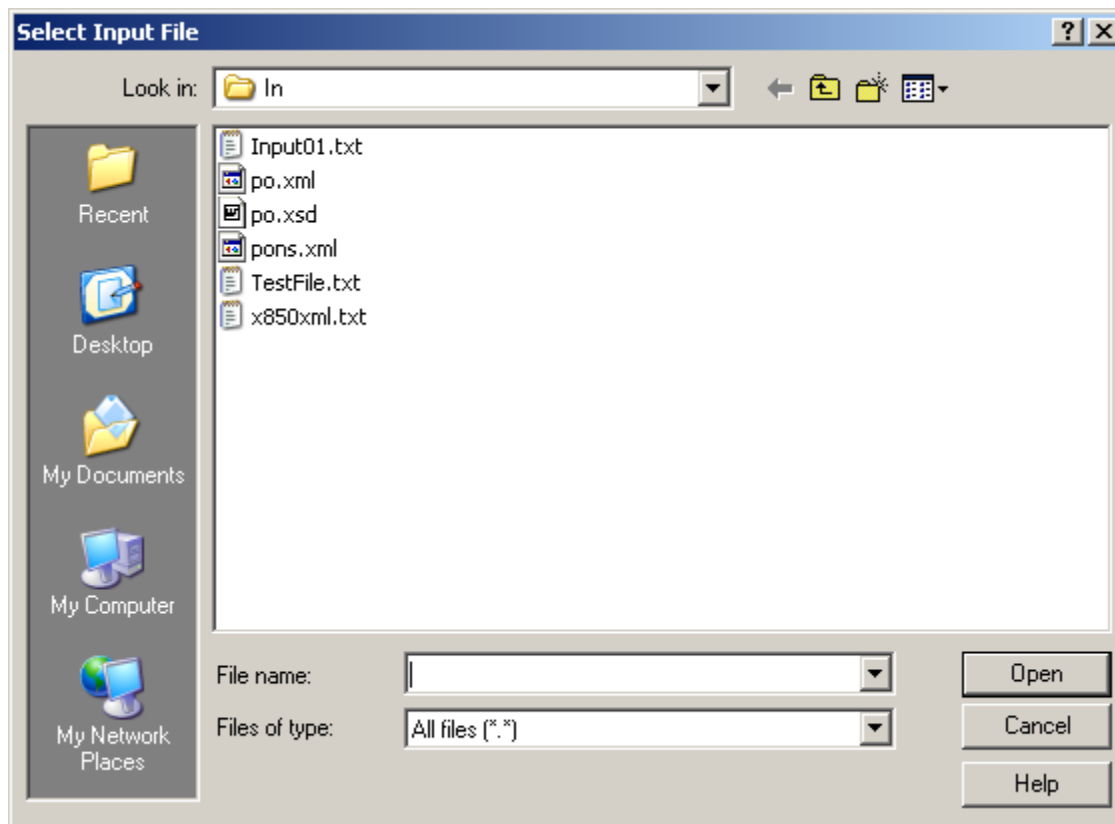
Select Input File Name and Location Dialog Box

Input Filename (Input List)

This column will contain the name of the input file(s) that you want to process. You add names to the list

by selecting the **Add Test Input** button  from the toolbar. The **Select input file name and location** dialog box appears to allow you to locate an input file from your directories. You can type a file name, choose from existing files using the drop-down list of the 10 most recently translated files, or select the **Browse** button. If you select the **Browse** button, the **Select Input File** dialog box appears.

You can change your selection or queue more input files by repeating the process. Once you make a selection, you can then view the input file by double-clicking on the file name. You can also select the column header to sort the list by file name.



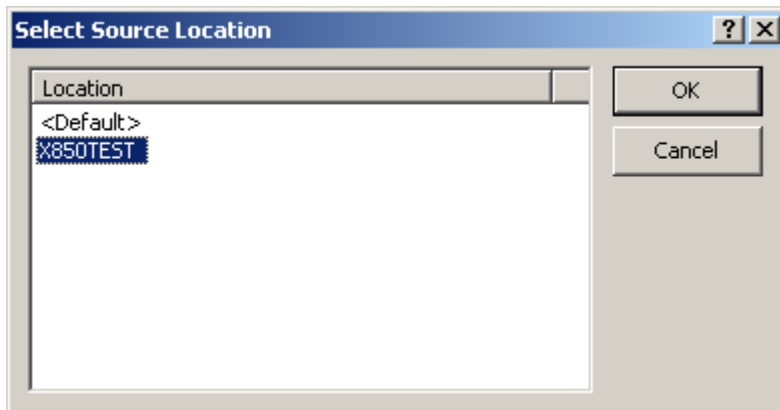
Select Input File Dialog Box

Location (Input List)



You add file names and locations to the list by selecting the **Add Test Input** button from the toolbar. The **Select input file name and location** dialog box appears to allow you to add a location from a list. You can type a location name, choose from existing location using the drop-down list of 10 most recently used locations, or select the **Browse** button. When you select the **Browse** button, the **Select Source Location** dialog box appears. This dialog lets you select from all locations defined in Partner Explorer. If you select the **Browse** button, the **Select Source Location** dialog box appears. Once you make a selection and close the dialog box, the location you selected will appear in the **Location** column.

When you are testing, this value must match the location in the **Locations/StdID** folder that contains the wrapper definition, or the standard will not be properly identified. If the incoming wrapper definition is associated with the <Default> location, you do not need to enter a source location here. You can leave it blank. Whatever you enter here will be displayed on the processing report as the source location for the input file.



Select Source Location

Bytes Read (Input List)

This indicates the number of bytes that have been read from the input file during processing. After processing, it will show the input file size. If this entry is blank, the file has not yet been run.

Status (Input List)

This is the current status of the input file. The possible statuses and their meaning are as follows:

Status	Description
Stopped	<ul style="list-style-type: none"> ▪ TRM has read the entire file ▪ Operator has stopped the translation
Running	<ul style="list-style-type: none"> ▪ TRM is currently running a test or a test has been paused
Aborted	<ul style="list-style-type: none"> ▪ The test ran and it aborted.

Output List (Test)

This box contains a list of all the files that the TRM creates when you process the files, including reports, output, and acknowledgments. The icons identify the file type. The system generates the file names based on a datetime stamp to ensure unique naming. They consist of 8 characters with an extension of .txt. The output file name will print on the processing reports if you have selected **Always Print Status** as a report option on the **General** tab of the Modify Options window. You can find the output files in the **OUT** subdirectory.

NOTE: There is an **Auto-delete Outputs** option on the **Test** menu that is cleared by default. When this option is checked, as soon as you exit the test module, the Workbench will delete all output files, including the processing report. This is to help you keep your system from accumulating test files unnecessarily. Do not select the option when you need to save the files.

The TRM creates two types of reports. The file wbrpt.txt contains the names of the global configuration files that it uses during processing. The second report, named after the input file, is created when actual processing starts. To view an output report, double-click the file name.

NOTE: The TRM reuses the processing report file names for each test run. If you want to save a processing report for later viewing, you must rename the report file before you rerun a test using the same input file name.

Output Filename (Output List)

This column will contain the names of the output files that the TRM generates. To view an output file, double-click the file name.

Location (Output List)

If the file is to be routed to a trading partner, this field identifies the recipient location name that the TRM generated during processing.

Bytes Written (Output List)

This indicates the number of bytes that have been written to the output file during processing. After processing, it shows the size of the output file.


Status (Output List)

This is the current status of the output file. The possible statuses and their meaning are as follows:

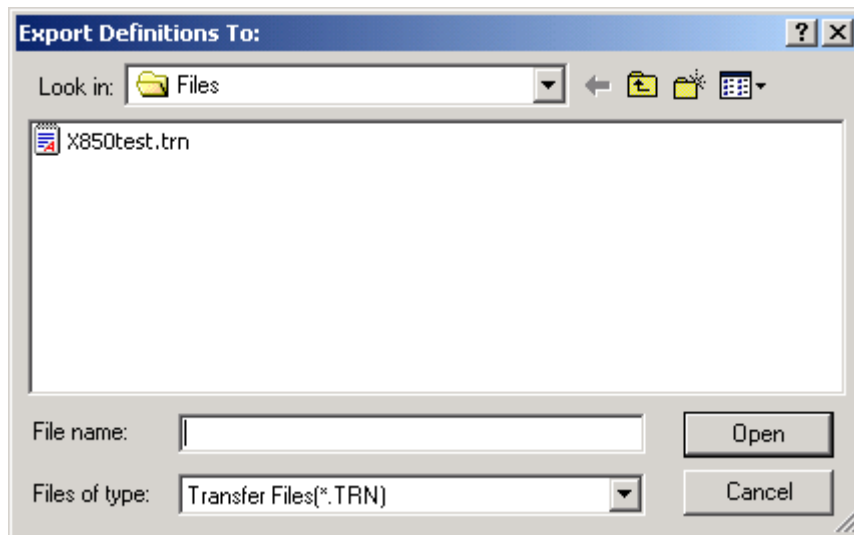
Status	Description
Created	The output has been created.
Complete	The output is complete. This happens only when the translation has stopped.
Aborted	The output has been aborted. In this case, the disk file has been erased.

Export Translation Definitions

When you have completed a successful translation, you can export all definitions required to run that

translation to a flat file. You do this by selecting the **Export Translation Definitions** button  from the toolbar. The **Export Definitions To** dialog box appears.

You enter the name of a file to which you will export the definitions. You can create a special subdirectory of your choice, or place them in the **Files** subdirectory. The Workbench will add an extension of *.trn*. The definitions can then be imported to another database using the Import command from the **Files** menu. For more information about the **Import** command, refer to the topic, *Import Command* (on page 501).



Export Definitions To Dialog Box

Procedures

You can perform the following tasks from the Test window.

To Test Configurations from the Workbench

The TRM may not generate a processing report if you do not have any errors, any PRINT statements in your map, or any items checked in the Report Options box on the **General** page of the Modify Options window. To specify the options for generating reports, refer to the topic *Procedures (Modify Options)* (on page 512).

Make sure the Test window is *open* (on page 762).




- 1 From the toolbar, choose the **Add Test Input** button.

The **Select input file name and location** dialog box appears.

- 2 Select the **Browse** button, and select your input file from the appropriate location.

The file name appears in the **Input Filename** box. If you had processed this file recently, you would be able to select it from the **Input Filename** list.

- 3 This step determines in which **Locations/StdID** folder the wrapper definition against which the file is to be matched is located. It can be in either the <Default> location folder or in a named folder. For information about the difference between the two choices, refer to the topic, *Task 6. Specifying the Source Location* (on page 300). Do one of the following:
 - If this input file is to be matched against a wrapper definition listed in the <Default> location folder, you can leave this field blank or select **<Default>**.
– or –
 - If this input is to be matched against a wrapper definition listed in a named location folder, you must enter that location name in the **Location** box. To do this, choose the **Browse** button, and select your location from the list. If you had processed this input recently, you would be able to select it from the **Location** list.
- 4 Select **OK**.
The name of the input file displays in the upper pane.
- 5 To queue more than one file, repeat steps 1-4.
- 6 From the upper pane, select the input file(s) you want to process.
If you select more than one file, then each file will be processed in order.
- 7 For easier reading, create the output so that each segment appears on a separate line. From the **Test** menu, choose **Force Text**.
- 8 From the toolbar, select the **Run Test** button .
If your test is successful, your report and output files appear in the lower pane.
- 9 To save your files after you have closed the Test window, make sure you have *not* selected the **Auto-delete Outputs** on the **Test** menu.
- 10 To review your files, refer to the topic, *To Review Test Files* (on page 772).

To Review Test Files

The TRM may not generate a processing report if you do not have any errors, any **PRINT** statements in your map, or any items checked in the **Report Options** box on the **General** page of the Modify Options window. To specify the options for generating reports, refer to the topic *Procedures (Modify Options)* (on page 512).

Make sure you have run a test as described in the task, *To Test Configurations from the Workbench* (on page 771).

Make sure the Test window is *open* (on page 762). To view files when the Test window is open or closed, do one of the following:


- When the Test window is open after running a test, double-click a file name to view the contents.
– or –

- To view these files after you have closed the test window:
 - Locate the processing report in the **RPT** subdirectory. It has the same file name as the input file. If you want to save the processing report, you should rename it, because the TRM will create a new report with the same name when you use the same input file for a subsequent test.
 - Locate the output file(s) in the **OUT** subdirectory. The TRM creates the output file names based on a datetime stamp, so there is no danger of generating duplicate output file names.

To Export Translation Definitions to a Flat File

You must already have just run a successful test translation from the Test window. If you still need to do this, refer to the instructions, *To Test Configurations from the Workbench* (on page 771).



- 1 From the toolbar, select the **Export Translation Definitions** button .
If this button is not available, you have not yet run a successful translation.
The **Export Definitions To** dialog box appears.
- 2 Type the subdirectory in which you want to save the file.
- 3 Type the name for your file that will contain the exported definitions.
- 4 Select **OK**.

An information box appears indicating that the Workbench is exporting the definitions.

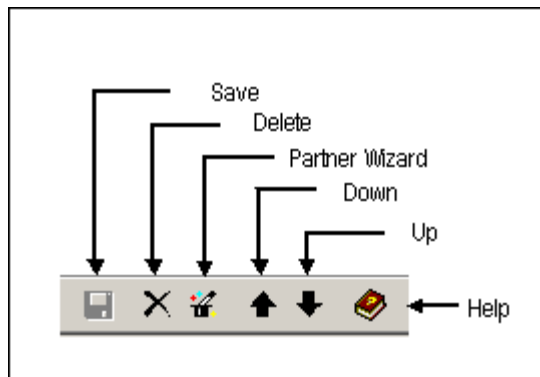
This page intentionally blank

Partner Explorer Reference

This reference information provides more information than the context-sensitive help.

Partner Explorer Toolbar

The following figure shows you a typical version of some additional icons that appear for the Partner Explorer. You can find more about these icons and their corresponding menu selections in the *General Reference* (on page 487).

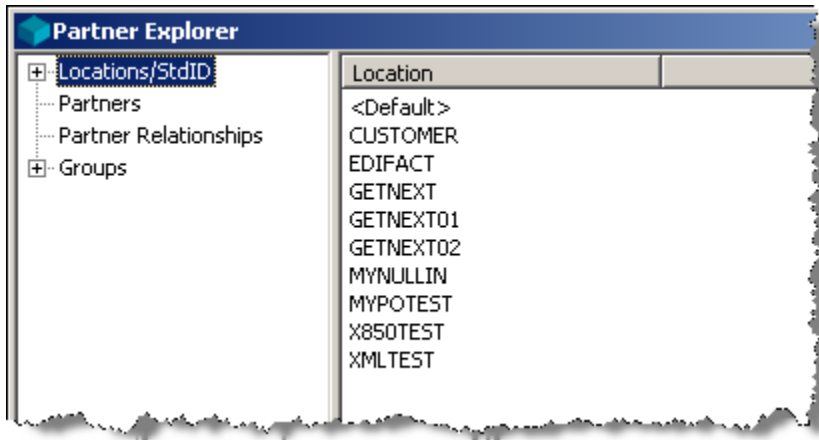


Partner Explorer Window

While the Data Explorer window allows you to specify data structure and processing options, the Partner Explorer allows you to control the processing of the data. From the Partner Explorer window, you can specify:

- How to identify incoming standards
- How to process the data based on known partners or on defaults for unknown partners

These configurations provide the rules by which the TRM uses the definitions you created using the Data Explorer.



Partner Explorer Window

Finding Things With Partner Explorer

To access all the definitions, expand the view of the folders in the left pane. When you find what you want in the right pane, you can double-click the entity and its definitions will display in a maintenance window.

All standard ID definitions are listed within their appropriate location folder. All partner definitions are listed within the **Partners** folder. These partners might also belong to a partner relationship or to a closed trade group.

Partner Explorer folders

The following table lists the Partner Explorer folders and what definitions you will find associated with each.

Folder	Description of Members
Locations/StdID	Each of the locations contains the standard ID wrappers that you want to use to identify the standard version of the incoming data, to parse the incoming wrapper, and to provide additional processing information.
Partners	Contains all partners that you have defined. Some might also belong to partner relationships or to groups.
Partner Relationships	Contains partners that will trade specifically with one another. Each partner definition also exists in the Partner folder. This relationship allows you to define special trading requirements for the relationship between these 2 partners.
Groups	Contains partners that will trade among themselves, but not with others who are not part of the group. Each partner definition is accessible from the Partner folder, not the group folder.
Group Name	Name of group created when you assigned a partner to a group.
Members	Contains a list of all sending partners defined for this group.
Recipients	Contains a list of recipient partners with at least one trade agreement profile assigned to this group.
Location/StdID	Contains a list of standard ID wrapper definitions with at least one trade agreement profile assigned to this group.

Groups

MW Translator maintains groups based on their use in partner definitions. You cannot add or delete groups from the Partner Explorer window.

You add and delete groups from the **Groups** tab of a Partner window. The Workbench creates the group when you assign a partner to be a member of a non-existent group. The Workbench deletes the group when the last member has been removed from the group. You may edit the group name and remove partners from groups from the partner window.

To Open the Partner Explorer Window

- From the **View** menu, choose **Partner Explorer**.

– or –



From the toolbar, choose the **Partner Explorer** button

Partner Window

The Partner window is one way you assemble the appropriate information to process an incoming file and route the output, that is, to control processing. You use this window to define profiles for your different trading partners, providing user IDs, routing information, and trade requirements and restrictions. These profiles can then become members of specific partner relationships or closed groups.

Procedures

These are some procedures you can perform from the Partner window.

To Add a Partner

Although partner definitions are not mandatory, they can provide an extra level of security during the matching phase. They are necessary when you have specific processing options that are different from other partners. For more information about partners, refer to the topic, *Defining Partners* (on page 243).

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the **Partners** folder.
- 2 From the **File** menu, select **Add**.
The **Add Partner** dialog box appears.
- 3 Type a name for this partner definition, and select **OK**.
The Partner window appears.
- 4 Select the **IDs** tab.
- 5 Type the appropriate ID values at the appropriate levels.
For more information about how this works, refer to the topic about the *function of the IDs during matching* (on page 53).
- 6 Select **OK**.

To Delete a Partner Profile

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the **Partners** folder.
- 2 In the right pane, select the partner you want to delete.
- 3 From the **File** menu, select **Delete**.

– or –



From the toolbar, select the **Delete** button

The **Confirm** dialog box appears.

- 4 Select **Yes**.

To Specify an Acknowledgment for a Partner

This allows you to define the necessary parameters for documents that will go back to the sender at the defined routing level. Typically, the partner profile for the sending partner triggers the generation of an acknowledgment. The level of the acknowledgment is defined as a document property for the acknowledgment definition.

Acknowledgments are defined at different levels for different standards. EDIFACT uses an interchange level acknowledgment. This means that there is potentially one acknowledgment generated for each incoming interchange. X12 uses a functional group level acknowledgment. This means that there is potentially one acknowledgment generated for each incoming functional group.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the **Partners** folder.
- 2 In the right pane, select the partner for which you want to specify an acknowledgment.
The Partner window appears.
- 3 Select the **Acknowledgments** tab.
- 4 Select the **Add** button.
The **Select Acknowledgment Profile** dialog box appears.
- 5 From the list of defined acknowledgment profiles, select the acknowledgment you want to send to this partner.
- 6 Select **OK** to add the acknowledgment to the partner's list.

To Specify a Trade Agreement Profile for a Partner

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the **Partners** folder.
- 2 In the right pane, select the partner for which you want to specify a trade agreement.
The Partner window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Select **Add**.
The **Select Trade Agreement Profile** dialog box appears.

- 5 If you want to associate this trade agreement with a closed group, type the group name in the **Closed Group Name** box.

This group is created under the **Groups** folder in the left pane.

- 6 From the list of defined trade agreement profiles, select the profile you want to use to process documents received by this partner.
- 7 Select **OK** to add the trade agreement to the partner's list.

To Create and Populate a Closed Trade Group

Make sure the Partner Explorer window is *open* (on page 777).

- 1 First, you must add a partner to a trade group, which also creates the group when it does not exist. Follow the procedure, *To Add a Sending Partner to a Closed Trade Group* (on page 781).
- 2 Repeat step 1 for every partner in the group.
All sending partners must be members of the group. The list of sending partners appears in the **Members** folder within the specific group folder.
- 3 From the Partner window, select **OK**.
- 4 Associate a trade agreement with a closed group for a particular recipient partner or for the standard ID. Use one of the following procedures:
 - *To Add a Trade Agreement Associated with a Recipient Partner to a Trade Group* (on page 782).
– or –
 - *To Add a Trade Agreement Associated with Standard ID to a Trade Group* (on page 781).

To Modify the Group for A Trade Agreement

Make sure the Partner Explorer window is *open* (on page 777).

- 1 Select the Partner or Standard ID configuration whose trade agreement you want to modify.
- 2 On the **Trade Agreements** tab, select the trade agreement from the list.
- 3 Right-click the trade agreement, and then select **Modify Group Name**.
– or –
Select the **Modify Group** button.
The **Enter Closed Group Name** dialog box appears.
- 4 Type the name, and select **OK**.
When the **Trade Agreements** tab reappears, the closed group name should appear for that trade agreement.
- 5 Select the **OK** or **Apply** button to complete the process.

To Add a Sending Partner to a Closed Trade Group

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the **Partners** folder.
- 2 In the right pane, double-click the partner you want to add to a trade group.
The Partner window appears.
- 3 Select the **Groups** tab.
- 4 Select **Add**.
The **Edit Groups** dialog box appears.
- 5 Type the name of the group.
If the group does not exist, the Workbench will create the group. This is the only way to create a group.
- 6 Select **OK** to add the partner to the group.

To Add a Trade Agreement Associated with Standard ID to a Trade Group

The trade agreement for a closed group may be attached to either a partner record, as the recipient, or to a standard ID record, as a default. You must determine with which entity the trade agreement should be associated, partner or standard ID. These instructions are for trade agreements that are already associated with standard ID records. If the trade agreement is not already associated with the standard ID, you will have to add it to the list.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane in the **Locations** folder, select the appropriate location.
- 2 In the right pane, double-click the standard ID whose trade agreement you want to associate with the group.
The Standard ID window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending on whether the trade agreement appears on the list:
 - If the trade agreement is on the list, select the trade agreement you want to add to the group.
– or –
 - If the trade agreement is not on the list, refer to the instructions, *To Specify a Trade Agreement Profile as a Default on Standard ID* (on page 859).
- 5 With the appropriate trade agreement selected, right-click the mouse, and then select **Modify Group Name**.
The **Enter Closed Group Name** dialog box appears.
- 6 Type the name, and select **OK**.

When the **Trade Agreements** tab reappears, the closed group name should appear for that trade agreement.

- 7 Select the **OK** or **Apply** button to complete the process.

To Add a Trade Agreement Associated with a Recipient Partner to a Trade Group

The trade agreement for a closed group may be attached to either a partner record, as the recipient, or to a standard ID record, as a default. You must determine with which entity the trade agreement should be associated, partner or standard ID. These instructions are for trade agreements that are already associated with partner records. If the trade agreement is not already associated with the partner, you will have to add it to the list.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane within the appropriate group folder, select the **Recipients** folder.
- 2 In the right pane, double-click the partner whose trade agreement you want to associate with the group.
The Partner window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending on whether the trade agreement appears on the list:
 - If the trade agreement is on the list, select the trade agreement you want to add to the group.
– or –
 - If the trade agreement is not on the list, refer to the instructions, *To Specify a Trade Agreement Profile for a Partner* (on page 779).
- 5 With the appropriate trade agreement selected, right-click the mouse, and then select **Modify Group Name**.
The **Enter Closed Group Name** dialog box appears.
- 6 Type the name, and select **OK**.
When the **Trade Agreements** tab reappears, the closed group name should appear for that trade agreement.
- 7 Select **OK**.

To Delete a Sending Partner from a Trade Group

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, expand the **Groups** folder, and then select **Members** from the appropriate group folder.
- 2 In the right pane, double-click the partner you want to delete from the group.
The Partner window appears.
- 3 From the Partner window, select the **Groups** tab.

- 4 From the list of groups on the **Groups** tab, select the group you want to remove from this partner's definition.
- 5 Select the **Remove** button.
- 6 Select **OK**.

To Delete a Trade Agreement Associated with Standard ID from a Trade Group

The trade agreement for a closed group may be attached to either a partner record, as the recipient, or to a Standard ID record, as a default. You will have to know with which entity the trade agreement is associated, Partner or Standard ID. You can find with which entity the trade agreement is associated on the processing report when you run a test. These instructions are for trade agreements associated with Standard ID records.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the appropriate location in the **Locations/StdID** folder.
- 2 In the right pane, double-click the standard ID whose trade agreement you want to delete from the group.
The Standard ID window appears.
- 3 Select the **Trade Agreements** tab.
- 4 From list of trade agreements on the **Trade Agreements** page, select the trade agreement you want to delete from the group.
- 5 Right-click the mouse, and from the menu, left-click **Modify Group Name**.
The **Enter Closed Group Name** dialog box appears.
- 6 Delete the name from the window, and select **OK**.
When the **Trade Agreements** page reappears, the closed group name should be blank for that trade agreement.
- 7 Select **OK**.

To Delete a Trade Agreement Associated with a Recipient Partner from a Trade Group

The trade agreement for a closed group may be attached to either a partner record, as the recipient, or to a Standard ID record, as a default. You will have to know with which entity the trade agreement is associated, Partner or Standard ID. You can find with which entity the trade agreement is associated on the processing report when you run a test. These instructions are for trade agreements associated with recipient partners.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane within the appropriate group, select the **Recipients** folder.
- 2 In the right pane, double-click the partner whose trade agreement you want to delete from the group.
The Partner window appears.

- 3 Select the **Trade Agreements** tab.
- 4 From list of trade agreements on the **Trade Agreements** tab, select the trade agreement you want to delete from the group.
- 5 Right-click the mouse, and when the **Modify Group Name** menu selection appears, left-click the mouse. The **Enter Closed Group Name** window appears.
- 6 Delete the name from the window, and select **OK**.
When the **Trade Agreements** page reappears, the closed group name should be blank for that trade agreement.
- 7 Select **OK**.

To Delete a Closed Trade Group

You cannot delete a group explicitly. The **Groups** folder is simply another view of what definitions are associated with the group. These definitions are maintained from their individual folders: **Partners** and **Locations/StdID**. When you remove all associations for the group, the Workbench deletes the group.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, expand the **Groups** folder, and select the group you want to have the Workbench delete.
- 2 Select the **Members** folder, and from the right pane select each partner and remove it from the group until there are no more partners in the folder.
For instructions, refer to the topic, *To Delete a Sending Partner from a Trade Group* (on page 782).
- 3 Select the **Recipients** folder, and from the right pane select each partner and remove it from the group until there are no more partners in the folder.
For instructions, refer to the topic, *To Delete a Trade Agreement Associated with a Recipient Partner from a Trade Group* (on page 783).
- 4 Select the **Locations/StdID** folder, and from the right pane select each standard ID/wrapper and remove it from the group until there are no more in the folder.
For instructions, refer to the topic, *To Delete a Trade Agreement Associated with Standard ID from a Trade Group* (on page 783).
When you have removed all associations with the closed group, the Workbench deletes the group.
- 5 If the group is still visible, press **F5** to refresh Partner Explorer.

To Add an Alias to a Partner

Remember that an alias for a partner works when a matching trade agreement profile output is also associated with that alias ID. Therefore, make sure that the *properties of the appropriate trade agreement outputs specify this alias ID* (on page 663).

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the **Partners** folder.
- 2 In the right pane, double-click the partner to which you want to add an alias.
The Partner window appears.
- 3 Select the **Alias** tab.
- 4 Select the **Add** button.
The **Enter Alias Partner** dialog box appears.
- 5 Type an alias type.
- 6 From the list of defined partners, select the profile you want to use.
- 7 Select **OK** to add the alias to the partner's list.

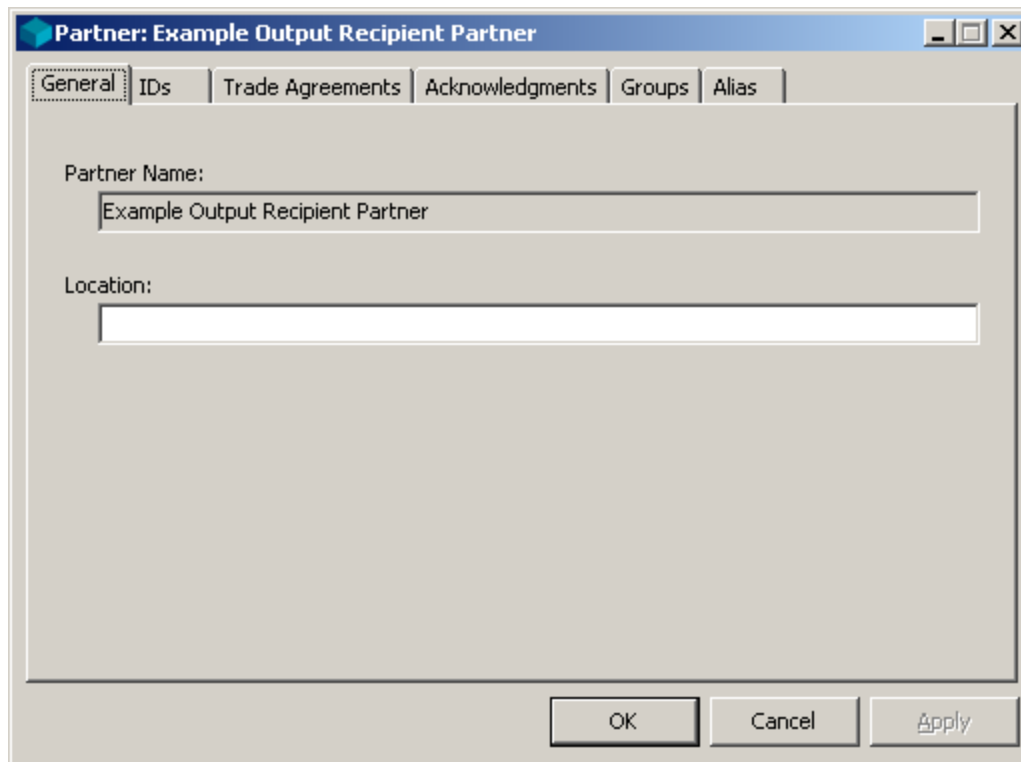
To Delete an Alias for a Partner Profile

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the **Partners** folder.
- 2 In the right pane, double-click the partner whose alias entry you want to delete.
The Partner window appears.
- 3 Select the **Alias** tab.
- 4 Select the alias that you want to delete, and select the **Remove** button.

(Partner) General Page

The **General** page displays the partner name and the location associated with this partner. You specify the name of the partner in the **Add Partner** dialog box.

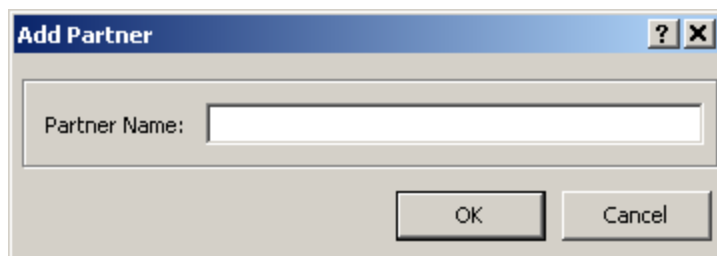


The screenshot shows a dialog box titled "Partner: Example Output Recipient Partner". It has a tabbed interface with the following tabs: "General" (selected), "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "General" tab contains two text input fields: "Partner Name:" with the value "Example Output Recipient Partner" and "Location:" which is empty. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

General Page (Partner Window)

Add Partner Dialog Box

The **Add Partner** dialog box appears when you **add** (on page 778) a partner.



The screenshot shows a dialog box titled "Add Partner". It has a "Partner Name:" label and an empty text input field. At the bottom of the dialog are two buttons: "OK" and "Cancel".

Add Partner Dialog Box

Partner Name

Enter a name for the trading partner. It may be from 1 to 40 alphanumeric characters.

Location

This is the location or address to which the TRM will route messages to this partner, unless routing is preempted by some other definition. It can be from 1 to 128 alphanumeric characters.

The location name is a value that messaging systems may use for routing. Whether an address can be passed to MessageWay depends on the process that sends the message, such as an adapter or service.

To route backward acknowledgments, the TRM uses the location value on the Partner window associated with the sender of the incoming document. To route other output, the TRM can use the location value for the receiver of the incoming document. However, other definitions might override or preempt this behavior. For more information, refer to the various topics on determining routing in the section, *How the TRM Processes Documents* (on page 46).

(Partner) IDs Page

The **IDs** page allows you to enter IDs for the first 2 levels of the 4 possible levels of wrappers. These internal fields should be defined on the Segment window for a particular element in one of the segments of the incoming wrapper. For an explanation of how the TRM uses these fields, refer to the topic, *Step 4. Look for Defined Partner IDs for the Sender and Recipient at this Level* (on page 53).

The **More** button allow you to enter ID information for levels 3 and 4.

The screenshot shows a dialog box titled "Partner: Example Output Recipient Partner". It has several tabs: "General", "IDs", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "IDs" tab is selected. The dialog is divided into two main sections: "Interchange" and "Functional Group". Each section contains three input fields: "ID", "Int ID", and "Int ID2", each followed by a "Qual:" field. In the "Interchange" section, the "ID" field contains the text "TESTREC". Below the "Functional Group" section is a "More..." button. At the bottom of the dialog are "OK", "Cancel", and "Apply" buttons.

IDs Page (Partner Window)

ID (IDs Page)

This field contains the primary partner ID. You can define up to 4 levels of sets of IDs. The levels should represent the various levels of wrappers for an incoming or outgoing document. You can enter from 1 to 35 alphanumeric characters.

For the TRM to relate the incoming wrapper to a partner in order to get the sender profile, this value must match the value in the incoming element assigned to the **Send Partner ID** internal field.

For the TRM to relate the incoming wrapper to a partner in order to get a recipient profile, this value must match the value in the incoming element assigned to the **Rec Partner ID** internal field.

Qual (IDs Page)

This field contains the partner ID qualifier. You can define up to 4 levels of sets of IDs. The levels should represent the various levels of wrappers for the incoming or outgoing document. You can enter from 1 to 4 alphanumeric characters.

For the TRM to relate the incoming wrapper to a partner in order to get the sender profile, this value must match the value in the incoming element assigned to the **Send Partner Qual** internal field.

For the TRM to relate the incoming wrapper to a partner in order to get a recipient profile, this value must match the value in the incoming element assigned to the **Rec Partner Qual** internal field.

Int ID (IDs Page)

This field contains the internal ID. You can define up to 4 levels of sets of IDs. The levels should represent the various levels of wrappers for the incoming or outgoing document. You can enter from 1 to 35 alphanumeric characters.

When the internal IDs are used in a particular wrapper, this value must match the value in the incoming element assigned to the **Send Partner Internal ID** internal field. This allows the TRM to relate the incoming wrapper to a partner in order to get the sender profile.

For the TRM to relate the incoming wrapper to a partner in order to get a recipient profile and a location to which it can route the output, this value must match the value in the incoming element assigned to the **Rec Partner Internal ID** internal field.

Int ID2 (IDs Page)

This field contains the internal sub-ID for this partner. You can define up to 4 levels of sets of IDs. The levels should represent the various levels of wrappers for the incoming or outgoing document. You can enter from 1 to 35 alphanumeric characters.

When internal ID2s are used in a particular wrapper this value must match the value in the incoming element assigned to the **Send Partner Internal ID2** internal field. This allows the TRM to relate the incoming wrapper to a partner in order to get the sender profile.

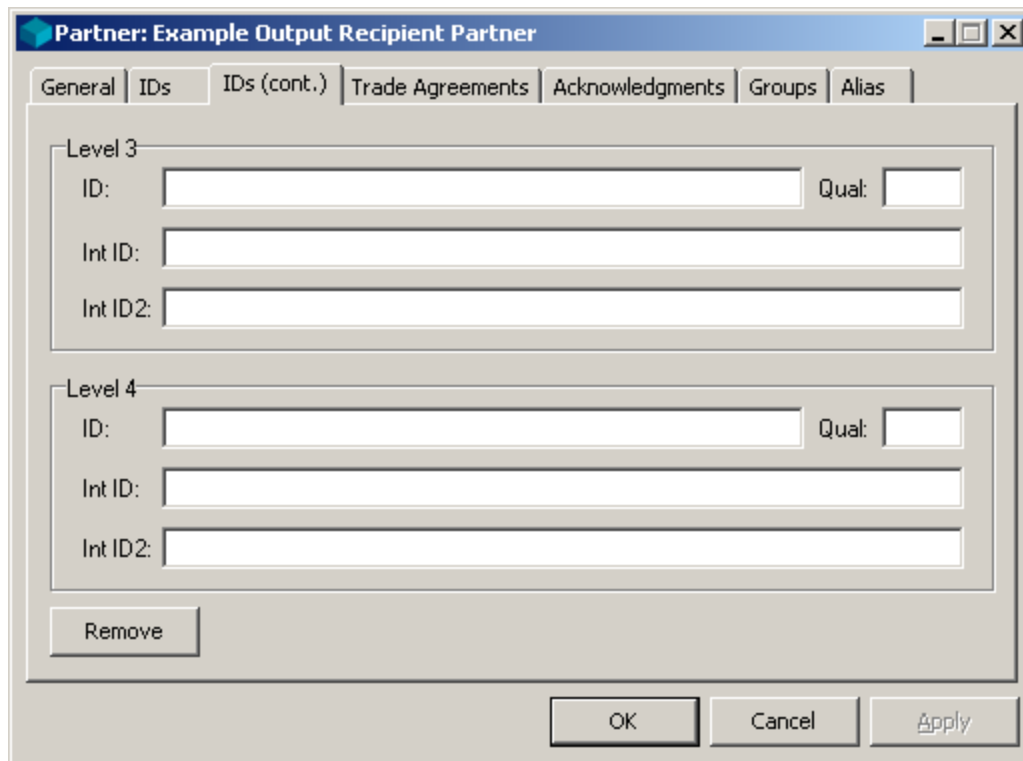
For the TRM to relate the incoming wrapper to a partner in order to get a recipient profile and a location to which it can route the output, this value must match the value in the incoming element assigned to the **Rec Partner Internal ID2** internal field.

More Button (IDs Page, Partner)

To enter or view ID values for wrapper levels 3 and 4, select the **More** button.

(Partner) IDs (cont.) Page

The **IDs (cont.)** page allows you to enter IDs for levels 3 and 4 of the 4 possible levels of wrappers.



The screenshot shows a window titled "Partner: Example Output Recipient Partner" with several tabs: "General", "IDs", "IDs (cont.)", "Trade Agreements", "Acknowledgments", "Groups", and "Alias". The "IDs (cont.)" tab is active. It contains two sections, "Level 3" and "Level 4". Each section has three input fields: "ID:", "Int ID:", and "Int ID2:". The "ID:" field also includes a "Qual:" dropdown menu. At the bottom left of the form area is a "Remove" button. At the bottom right are "OK", "Cancel", and "Apply" buttons.

IDs (cont.) Page (Partner Window)

ID (IDs (cont.) Page)

This field contains the partner ID. You can define up to 4 levels of sets of IDs. The levels should represent the various levels of wrappers for an incoming or outgoing document. You can enter from 1 to 35 alphanumeric characters.

For the TRM to relate the incoming wrapper to a partner in order to get the sender profile, this value must match the value in the incoming element assigned to the **Send Partner ID** internal field.

For the TRM to relate the incoming wrapper to a partner in order to get a recipient profile, this value must match the value in the incoming element assigned to the **Rec Partner ID** internal field.

These internal fields should be defined on the Segment window for a particular element in one of the segments of the incoming wrapper.

Qual (IDs (cont.) Page)

This field contains the partner ID qualifier. You can define up to 4 levels of sets of IDs. The levels should represent the various levels of wrappers for the incoming or outgoing document. You can enter from 1 to 4 alphanumeric characters.

For the TRM to relate the incoming wrapper to a partner in order to get the sender profile, this value must match the value in the incoming element assigned to the **Send Partner Qual** internal field.

For the TRM to relate the incoming wrapper to a partner in order to get a recipient profile, this value must match the value in the incoming element assigned to the **Rec Partner Qual** internal field.

These internal fields should be defined on the Segment window for a particular element in one of the segments of the incoming wrapper.

Int ID (IDs (cont.) Page)

This field contains the internal ID. You can define up to 4 levels of sets of IDs. The levels should represent the various levels of wrappers for the incoming or outgoing document. You can enter from 1 to 35 alphanumeric characters.

When internal IDs are used in a particular wrapper, this value must match the value in the incoming element assigned to the **Send Partner Internal ID** internal field. This allows the TRM to relate the incoming wrapper to a partner in order to get the sender profile.

For the TRM to relate the incoming wrapper to a partner in order to get a recipient profile and a location to which it can route the output, this value must match the value in the incoming element assigned to the **Rec Partner Internal ID** internal field.

These internal fields should be assigned on the Segment window for a particular element in one of the segments of the incoming wrapper.

Int ID2 (IDs (cont.) Page)

This field contains the internal sub-ID for this partner. You can define up to 4 levels of sets of IDs. The levels should represent the various levels of wrappers for the incoming or outgoing document. You can enter from 1 to 35 alphanumeric characters.

When internal ID2s are used in a particular wrapper, this value must match the value in the incoming element assigned to the **Send Partner Internal ID2** internal field. This allows the TRM to relate the incoming wrapper to a partner in order to get the sender profile.

For the TRM to relate the incoming wrapper to a partner in order to get a recipient profile and a location to which it can route the output, this value must match the value in the incoming element assigned to the **Rec Partner Internal ID2** internal field.

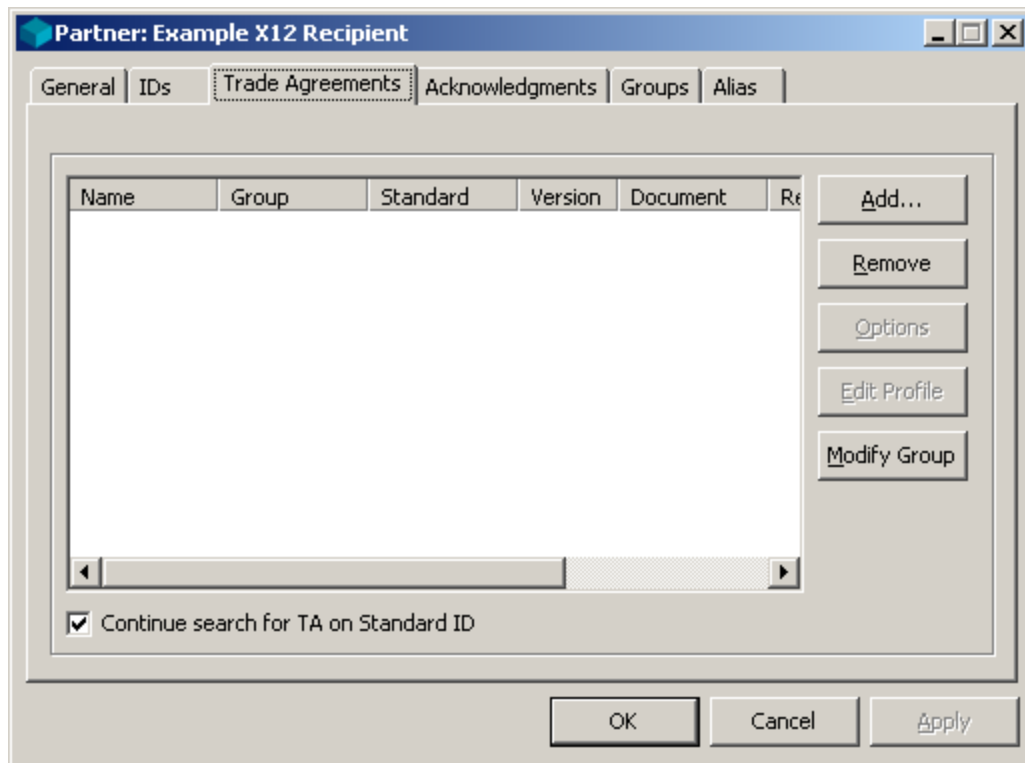
These internal fields should be assigned on the Segment window for a particular element in one of the segments of the incoming wrapper.

Remove Button

To remove ID values for wrapper levels 3 and 4, select the **Remove** button.

(Partner) Trade Agreements Page

The **Trade Agreements** page identifies those trade agreements this partner may use. This is also where you associate trade agreements for recipient partners with closed trade groups. If there are no values here, the TRM has a strategy to find alternative trade agreements. For a discussion of this strategy, refer to the topic *Step 9. Find a Trade Agreement Profile* (on page 61).

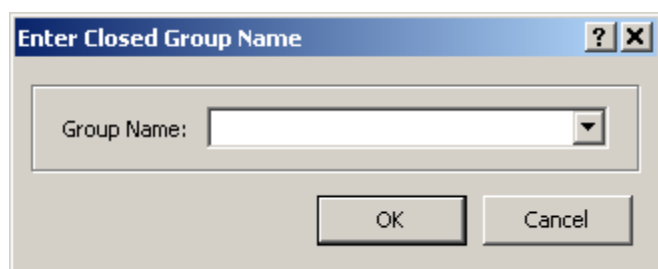


Trade Agreements Page (Partner Window)

Trade Agreements List

You can have multiple trade agreement profiles for a given partner profile. This is where you identify which documents this partner will receive. At runtime, the TRM will select the appropriate profile by matching the input data with the values that are listed here for the various columns.

If you want to assign a trade agreement already on the list to a group or change the assignment of a trade agreement to a group, select the **Modify Group** button. The **Enter Closed Group Name** dialog box appears.



Enter Closed Group Name Dialog Box

Name (Trade Agreements List)

The name uniquely identifies the trade agreement profile. It can contain 1 to 32 alphanumeric characters. It can have one or more output trade agreements defined for it, each of which can have multiple translation outputs based on different maps.

Group (Trade Agreements List)

Enter a new group or select an existing group with which you want to associate this trade agreement. Trade agreements that are not associated with a group take precedence over those that are associated with a group. The TRM may select this trade agreement, assuming other fields also match, only when it has not already found a matching trade agreement on the list that does not belong to a group. The group must match one of the groups listed on the **Groups** tab of the sending partner's profile.

Standard (Trade Agreements List)

This is the standard of the incoming document. The value here must match the standard identified during the standard identification process.

Version (Trade Agreements List)

This is the interchange version of the standard identified in the incoming wrapper set.

IMPORTANT: When the trade agreement profile displays a value here, it must match the value received in the element that is assigned to the internal field, **Interchange Version**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Interchange Version**, then this internal field will be null, and will not match the value in this **Version** column.

Document (Trade Agreements List)

This is the document ID of the incoming document.

IMPORTANT: When the trade agreement profile displays a value here, it must match the value received in the element that is assigned to the internal field, **Document Id**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Document Id**, then this internal field will be null, and will not match the value in this **Document** column.

Release (Trade Agreements List)

This is the release of the standard identified in the incoming wrapper set.

IMPORTANT: When the trade agreement profile displays a value here, it must match the value received in the element that is assigned to the internal field, **Release**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Release**, then this internal field will be null, and will not match the value in this **Release** column.

Agency (Trade Agreements List)

This is the agency identified in the incoming wrapper set.

IMPORTANT: When the trade agreement profile displays a value here, it must match the value received in the element that is assigned to the internal field, **Agency**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Agency**, then this internal field will be null, and will not match the value in this **Agency** column.

Assoc (Trade Agreements List)

This is the association identified in the incoming wrapper set.

IMPORTANT: When the trade agreement profile displays a value here, it must match the value received in the element that is assigned to the internal field, **Association Code**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Association Code**, then this internal field will be null, and will not match the value in this **Assoc** column.

Continue Search for TA on Standard ID

This check box allows you to specify that you want to search for other trade agreements if there are none on this list or none on the list match. This gives you explicit control over default processing.

If you want to search this list but not continue if there is no match, you should make sure this box is not checked.

If you want to search the next trade agreement list, which would be for the inbound Standard ID and wrapper identified during the standard identification, check this box.

Add Button

To add trade agreements to this partner profile, you select the **Add** button, and the **Select Trade Agreement Profile** dialog box appears.

Remove Button

The **Remove** button allows you to remove a trade agreement profile from the list. This does not delete the profile, it only makes it unavailable as a possible choice for this partner.

Modify Group Button

The **Modify Group** button allows you to modify the value in the **Group** column. You can change the association of the trade agreement with a group, add an existing trade agreement to a group, or delete a trade agreement from a group.

Options Button

The **Options** button displays the **Trade Agreement Options** window, which allows you to specify alternative IDs, location addresses, and service characters for delimited standards, for the output created using the selected trade agreement. It also allows you to specify a true value for test and acknowledgment-requested elements in the wrapper. In cases where the output sequences of the Trade Agreement profile, which is listed in the **Trade Agreements** folder of Data Explorer, do not match those of the Trade Agreement Options, the **Adjust Output Sequences** dialog box appears to allow you to fix the problem.

Adjust Output Sequences Dialog Box

This dialog box appears when you add or delete outputs to a trade agreement profile (Data Explorer) after you have referenced these profiles in a partner, partner relationship, or standard ID Trade Agreement list. As a result, there will be a different number of outputs in the trade agreement profile versus the options, or the sequence numbers of the outputs might not match those of the options. Since the options are where you specify alternative IDs and locations, this could be a problem. When you access the **Options** button, the Workbench recognizes this, and allows you to adjust the sequences in one of two ways:

- The **Adjust the output sequence(s) and keep the original properties** option saves any previously entered options, but at the risk that they were originally intended for a different output.
- The **Adjust the output sequence(s) and DELETE the original properties** option is the safer of the two. It removes all previous trade agreement options and allows the user to re-enter these options correctly.



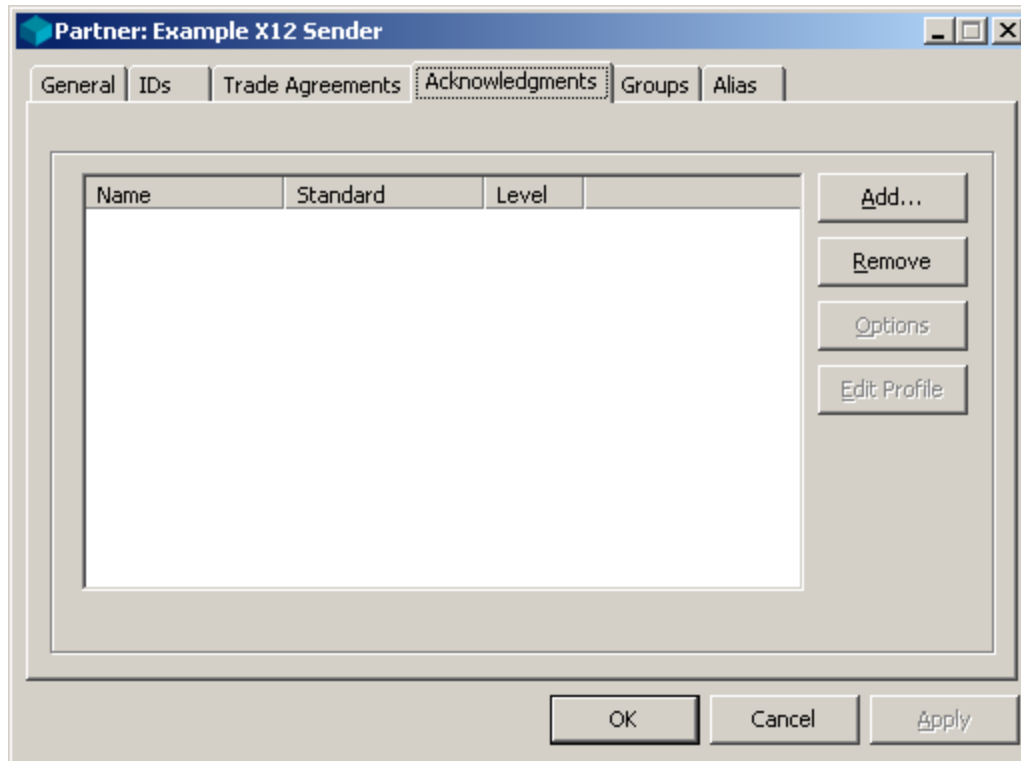
Adjust Output Sequences Dialog Box

Edit Profile Button (Partner Window, Trade Agreements Page)

The **Edit Profile** button is active only for the Workbench. It allows you to change the definition of the selected trade agreement profile. For more information, refer to the description of the **Trade Agreement Profile window** (on page 652).

(Partner) Acknowledgments Page

The **Acknowledgments** page identifies the acknowledgments that this trading partner can receive.



Acknowledgments Page (Partner Window)

Acknowledgments List

You can have multiple acknowledgment profiles for a given partner profile. This is where you identify which ones are associated with this partner. The TRM will generate any acknowledgments on the list that match the wrapper for the identified inbound standard that is the subject of the acknowledgment.

Name (Acknowledgments List)

The name uniquely identifies the acknowledgment profile. The acknowledgment profile identifies which acknowledgments are to be returned to this partner when this partner is the sender. You can add and remove defined acknowledgment profiles for this partner list by using the **Add** and **Remove** buttons. You cannot change the profiles from here.

Standard (Acknowledgments Listbox)

The wrapper of the incoming document identified during standard identification must match the standard and version shown here from the acknowledgment profile.

Level (Acknowledgments List)

The **Level** identifies the level of wrapper to which the acknowledgment responds. It was assigned when the acknowledgment was defined.

There may be one acknowledgment for each occurrence of that level wrapper. The options are as follows:

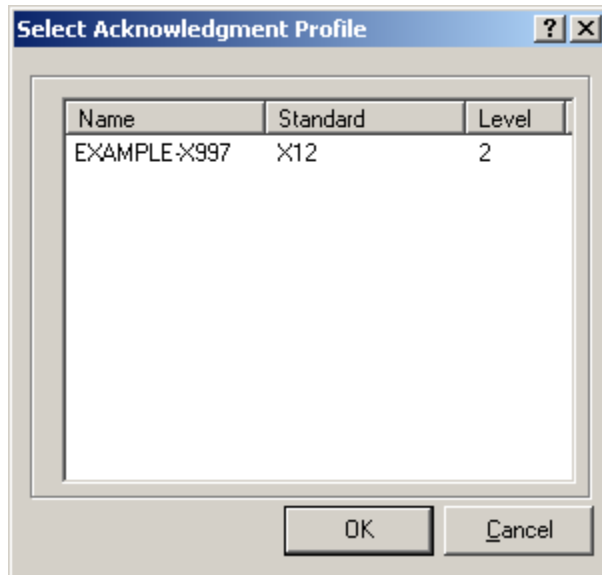
Level	Description
0	Acknowledgment level not applicable. This is used for documents that do not receive acknowledgments themselves, and are described as acknowledgments, but that in fact provide some other function, such as AUTACK for EDIFACT.
1	Interchange level acknowledgments are in response to all functional groups, if any, and all documents that the interchange contains. There will be one acknowledgment for each interchange level wrapper.
2	Functional group level acknowledgments are in response to all documents within a functional group. There will be one acknowledgment for each functional group level wrapper.

Add Button (Partner Window, Acknowledgments Page)

You may add pre-defined acknowledgments to this partner list by selecting the **Add** button. This displays the **Select Acknowledgment Profile** dialog box.

Select Acknowledgment Profile Dialog Box (Partner Window, Acknowledgments Page)

You select a pre-defined acknowledgment from the **Select Acknowledgment Profiles** dialog box. The list displays information that should allow you to distinguish between profiles, such as the profile name, the standard to which it belongs, and the level of the acknowledgment.



Select Acknowledgment Profile Dialog Box (Acknowledgments Page)

Remove Button (Partner Window, Acknowledgments Page)

The **Remove** button allows you to remove an acknowledgment profile for the partner. It does not delete the profile. This means that this partner no longer accepts this acknowledgment.

Options Button (Partner Window, Acknowledgments Page)

The **Options** button allows you to enter alternative information for the trading partner in the outbound acknowledgment. This includes a different sender and receiver location IDs. By default for delimited standards, the acknowledgment uses the same service characters as the input document that is the subject of the acknowledgment. If you don't want to use the service characters that were used in the input data, you can specify different ones here.

Edit Profile Button (Partner Window, Acknowledgments Page)

The **Edit Profile** button allows you to change the trade agreement profile information.

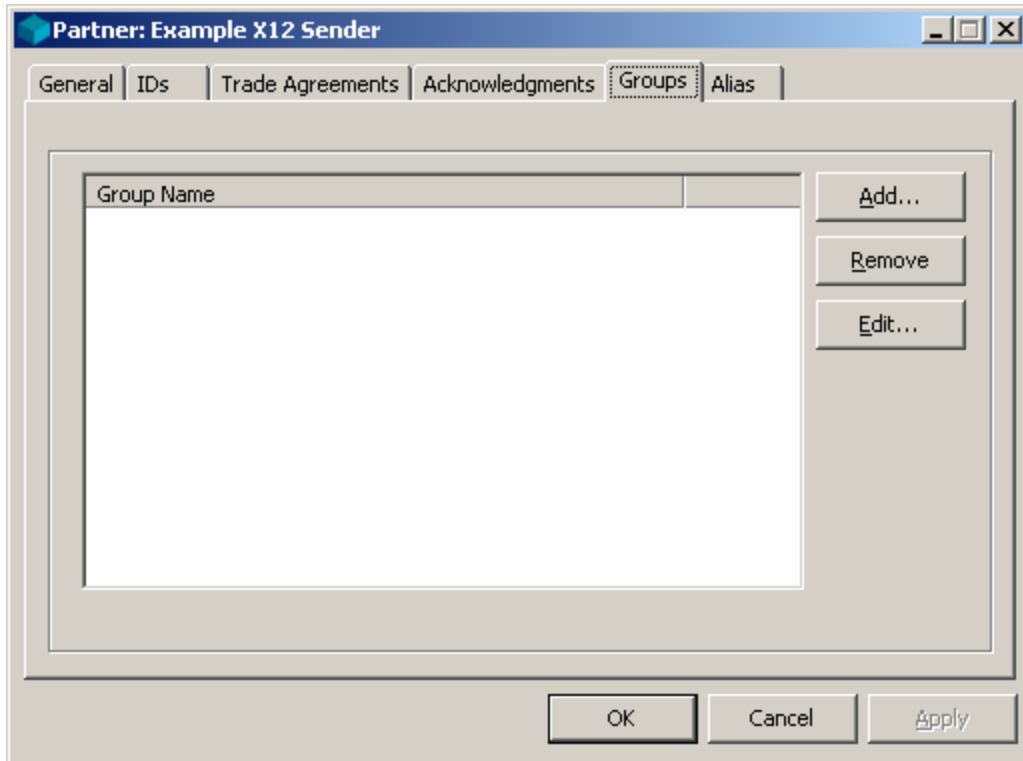
(Partner) Groups Page

The **Groups** page identifies those groups to which this sending partner belongs. This partner name appears in the **Members** folder when you add a group to this list. This list of groups is used to select trade agreements for recipient partners that belong to a closed group.

IMPORTANT: Trade agreements that belong to groups may be matched only when trade agreements that do not belong to a group fail to match.

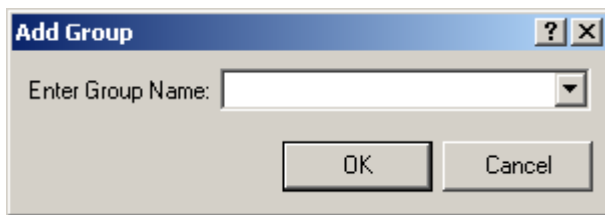
First the TRM looks for a trade agreement for the recipient partner that does not belong to a group. If it fails to find a matching trade agreement that does not belong to a group, then it checks for trade agreements for the recipient partner that do belong to groups.

The TRM uses the list of groups for the sender to match against the list of trade agreements that belong to one of these groups. It will then choose the first trade agreement that matches.

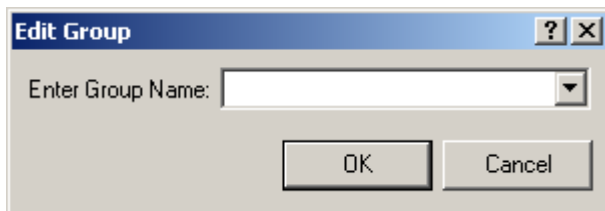


Groups Page (Partner Window)

When you add the first group and choose **Add**, the **Add Group** dialog box appears.



When groups are already listed and you choose **Add** or **Edit**, the **Edit Group** dialog box appears.



Groups List box

This list box allows you to add this sending partner to or remove this partner from a closed trade group using the **Add** and **Remove** buttons. MW Translator maintains groups based on partner definitions. MW Translator creates groups when you add a partner to a group that does not yet exist. The name of the partner then appears in the **Members** folder for that group. MW Translator removes groups when the last member is deleted from the group.

Add Button (Partner Window, Groups Page)

The **Add** button allows you to add this partner to a selected closed group. If the group does not exist, MW Translator creates it and places this member in the group's **Members** folder.

Remove Button (Partner Window, Groups Page)

The **Remove** button allows you to remove this partner from a selected closed group. When all senders in the **Members** folder and all recipients in the **Recipients** folder have been removed from this group, MW Translator deletes the group.

Edit Button (Partner Window, Groups Page)

The **Edit** button allows you to change the selected group name to which this partner belongs.

(Partner) Alias Page

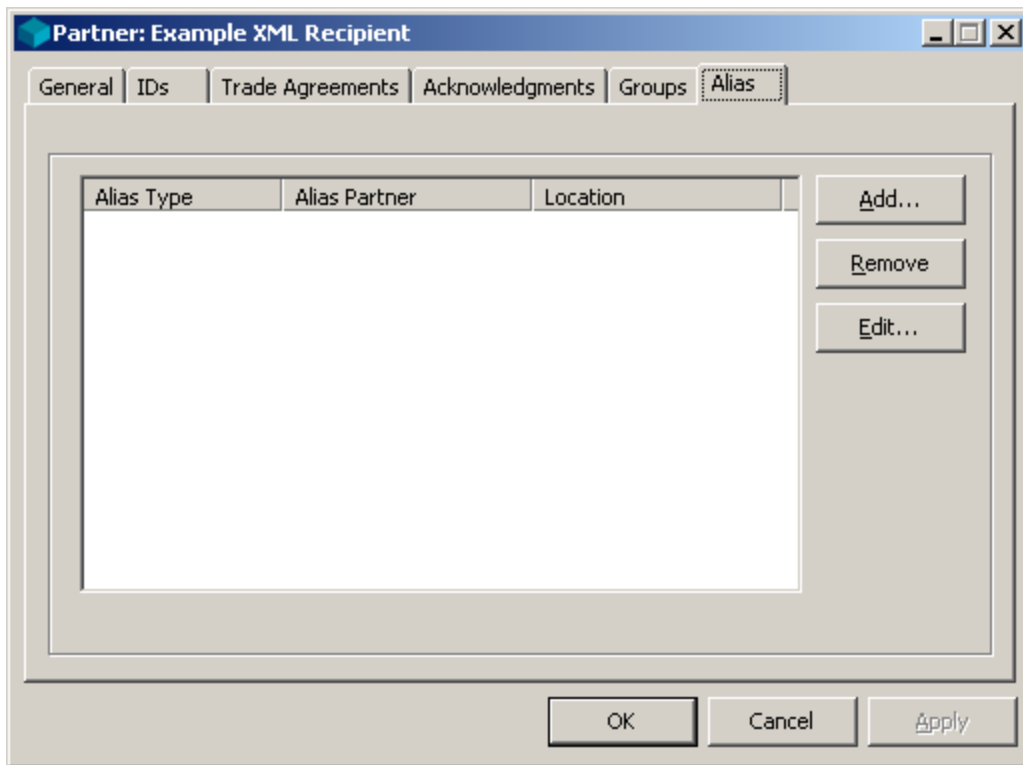
The **Alias** page identifies those aliases for this partner. They provide alternative IDs and/or locations. If alternative IDs or locations are missing in the Trade Agreement Properties for the sender or receiver, and if an alias type exists for the trade agreement output (Trade Agreement Output Properties window), the TRM will attempt to use aliases. Aliases are most useful when you have either a sender or receiver ID from the input but need to supply both sender and receiver IDs for the output.

The effect of using aliases for output is the same as if you were to use Partner Trade Agreement Options, which also provide alternative sender and receiver IDs and source and destination locations. It makes sense to use Partner Trade Agreement Options if you know the sender and recipient IDs at run time. However, if you are configuring open and closed trade agreements (TAs), you do not know both partners at configuration time. When configuring an open trade agreement, you would typically know the recipient ID but not the sender, and when configuring a closed trade agreement you may only know the sender ID. In addition, an open or closed TA may affect many different partners. As a result, you cannot directly specify the outbound ID, as you do with partner trade agreement properties, but instead you must define a mapping between inbound partners and outbound partners. Using a special wildcard feature in aliases, you are able to supply an alternative recipient ID when you only have an incoming sender ID and an alternative sender ID when you only have an incoming recipient ID. The TRM selects the appropriate IDs at runtime.

IMPORTANT: Trade agreement options take precedence over aliases. If you want to use aliases for a partner, make sure you do not also define trade agreement options for that partner.

If you do not want to use aliases, you can leave this page blank.

NOTE: You cannot use aliases for partner relationships. You can only use aliases for partners in open trade or closed groups. Aliases do not affect acknowledgments.



Alias Page (Partner Window)

Alias List

This is where you add and remove alternative partner definitions for the trading partner. These alternative definitions are called aliases, which indirectly link the listed partners with the current partner. This allows you to use a different partner definition for such entities as a name and location for the output rather than those currently represented on the **Partner** and **IDs** pages. You can only select existing partner records as aliases. You cannot change the IDs, but you can provide a different, overriding location.

If at configuration time you only know a sender or receiver ID, but want to supply values for both the sender and receiver in the output, you can use a wildcard character. You must add a question mark (?) to the alias type on the **Alias** page of the Partner Trade Agreement Output Properties window. Then in your partner record, you add an **S** to the alias type for the sending partner and an **R** to the alias type for the recipient partner. For more information about this feature, refer to the topics *Step 13. Determine IDs for Outbound Partners* (on page 74) and *Step 14. Determine Routing and Miscellaneous Information (Translation)* (on page 91).

Enter Alias Partner Dialog Box (Alias Page)

Alias Type (Alias List)

The Alias Type field contains the value that identifies this partner configuration at this level. The same partner might have different partner definitions, such as, one for proprietary standards (APPL1) and one for X12 standards (X12). The aliases are linked to identify the various definitions that belong to one partner. With aliases, the TRM can use different sender and recipient IDs and destination location for the output document.

Alias Partner (Alias List)

This name identifies the alternative partner profile that belongs to the alias type. You must select the partner from a list of existing partner definitions.

Location (Alias List)

This is the alternative location defined for the alternative partner profile. This is where the output will go, instead of going to the location of the partner profile identified for the recipient. This box displays the location, if one exists, for the partner you have chosen. You can override this location by entering a new value.

Add Button

The **Add** button allows you to add aliases to the list for this partner.

Remove Button

The **Remove** button allows you to remove aliases from the list for this partner.

Edit Button

The **Edit** button allows you to change the alias information you see in the list box.

Partner Trade Agreement Options Window

This window allows you to define values that would be used as alternative values when the TRM generates the outbound document. By default, the TRM uses the same IDs for the outbound wrappers and the same location for the outbound documents that belong to the sending and recipient partners identified in the incoming wrapper. Since trade agreements can have multiple outputs, you can identify to which of the outputs this alternative information applies.

This provides a direct cross-reference between partner profiles, unlike aliasing, which is an indirect cross-reference.

Output

Trade agreements can have multiple outputs. This selection box allows you to specify the output to which these alternative IDs, location, and miscellaneous configurations will apply. It also displays the sequential order of the output, if there are multiple from a given trade agreement.

Procedures (Partner Trade Agreement Window Options)

The procedures associated with trade agreement properties do not affect backward acknowledgments.

To Specify Alternative Information in Outbound Wrappers for a Partner

As a default strategy, the TRM uses the same IDs and location for the outbound documents that belong to the sending and recipient partners identified in the incoming wrapper. You can specify alternative IDs and destination location in one of two ways: by using aliases or by specifying values directly. This procedure specifies values directly.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partners**.
- 2 In the right pane, double-click the recipient partner for which you want to provide alternative partner ID(s) in an outbound wrapper or an alternative destination location.
The Partner window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending on whether the trade agreement appears on the list:
 - If the trade agreement appears on the list, select it, and then select the **Options** button.
The **Partner Trade Agreement Options** window appears.
– or –
 - If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to the topic, *To Specify a Trade Agreement Profile for a Partner* (on page 779).
- 5 From the list, select an existing partner to use its ID(s) and location.
– or –
Enter a new partner with new IDs and location. This creates a partner in the **Partners** folder.
- 6 Select **OK**.

To Generate a Document in Its Own Interchange for a Partner

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partners**.
- 2 In the right pane, double-click the recipient partner on whose trade agreement you want to specify the unconditional break.
The Partner window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending on whether the trade agreement appears on the list:
 - If the trade agreement appears on the list, select it and then choose the **Options** button.

The **Partner Trade Agreement Options** window appears.

– or –

- If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to the topic, *To Specify a Trade Agreement Profile for a Partner* (on page 779).

5 Select the **Misc** tab, and then select the **Document Level Break** check box.

6 Select **OK**.

To Specify When a Document Requires an Acknowledgment for a Partner

You can keep track of documents that require responding acknowledgments using document reconciliation. To specify which documents expect acknowledgments, follow this procedure.

NOTE: You must also select **Enable Reconciliation** on the **MessageWay Settings** tab of the Operator program.

Make sure the Partner Explorer window is *open* (on page 777).

1 In the left pane, select **Partners**.

2 In the right pane, double-click the recipient partner.

The Partner window appears.

3 Select the **Trade Agreements** tab.

4 Do one of the following, depending on whether the trade agreement appears on the list:

- If the trade agreement appears on the list, select it and then choose the **Options** button.

The **Partner Trade Agreement Options** window appears.

– or –

- If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to the topic, *To Specify a Trade Agreement Profile for a Partner* (on page 779).

5 Select the **Misc** tab, and then select the **Acknowledgment Expected** check box.

6 Select **OK**.

(Partner Trade Agreement Options) Sending Partner Page

The **Sending Partner** page of the outbound Trade Agreement Options window specifies levels 1 (Interchange) and 2 (functional group) of the 4 possible levels for the sending partner. If you do not want to specify alternative sending partner information, you can leave this page blank. It is valid for outbound documents only, not for acknowledgments.

Partner Trade Agreement Options: EXAMPLE-X850 ? X

Output: 1,EXAMPLE,1,FPO 1 of 1

Sending Partner Recipient Partner Misc

Partner Name:

Location:

Override Location:

Interchange

ID: Qual:

Int ID: Int ID2:

Functional Group

ID: Qual:

Int ID: Int ID2:

OK Cancel Apply

Sending Partner Page (Partner Trade Agreement Options Window)

Partner Name (Sending Partner)

This identifies a different partner as the sender, whose definitions will be used for the output wrapper. Use this to override the outbound sending partner. You can select a partner name from the existing definitions, or type a new name, and add information to create a new partner.

Location (Sending Partner)

This is the source location defined for the sending partner.

Override Location

Enter a different location here to override the location listed for this partner.

ID (Sending Partner)

This field contains the partner ID for the outbound sending partner. For more information, refer to the topic, *ID (IDs Page)* (on page 789).

Qual (Sending Partner)

This field contains the qualifier for the outbound sending partner. For more information, refer to the topic, *Qual (IDs Page)* (on page 789).

Int ID (Sending Partner)

This field contains the internal ID for the outbound sending partner. For more information, refer to the topic, *Int ID (IDs Page)* (on page 789).

Int ID2 (Sending Partner)

This field contains the internal sub-ID for the outbound sending partner. For more information refer to the topic, *Int ID2 (IDs Page)* (on page 790).

(Partner Trade Agreement Options) Recipient Partner Page

This page allows you to specify alternative partner IDs and location for levels 1 (Interchange) and 2 (functional group) of the 4 possible levels for the recipient partner. It is valid for outbound documents only, not for acknowledgments.

Partner Trade Agreement Options: EXAMPLE-X850 [?] [X]

Output: 1,EXAMPLE,1,FPO 1 of 1

Sending Partner | **Recipient Partner** | Misc

Partner Name: []

Location: []

Override Location: []

Interchange

ID: [] Qual: []

Int ID: [] Int ID2: []

Functional Group

ID: [] Qual: []

Int ID: [] Int ID2: []

[OK] [Cancel] [Apply]

Recipient Partner Page (Partner Trade Agreement Options Window)

Partner Name (Recipient Partner)

This identifies a different partner as the recipient, whose definitions will be used for the output wrapper. Use this to override the outbound recipient partner. You can select a partner name from the existing definitions, or type a new name, and add information to create a new partner.

Location (Recipient Partner)

This is the location to which the TRM will route output destined for this recipient partner.

Override Location

Enter a different location here to override the location listed for this partner.

ID (Recipient Partner)

This field contains the partner ID for the outbound recipient partner. For more information refer to the topic, *ID (IDs Page)* (on page 789).

Qual (Recipient Partner)

This field contains the partner ID qualifier for the outbound recipient partner. For more information refer to the topic, *Qual (IDs Page)* (on page 789).

Int ID (Recipient Partner)

This field contains the internal ID for the outbound recipient partner. For more information refer to the topic, *Int ID (IDs Page)* (on page 789).

Int ID2 (Recipient Partner)

This field contains the internal sub-ID for the outbound recipient partner. For more information, refer to the topic, *Int ID2 (IDs Page)* (on page 790).

(Partner Trade Agreement Options) Misc Page

The **Miscellaneous** page allows you to specify the **Request Acknowledgment** and **Test** values in the outbound wrapper. You may also specify delimiters for the document that would override those specified for the outbound wrapper. You may also control document breaks in the output stream, specify that documents created with this trade agreement be marked to expect acknowledgments for reconciliation purposes and have MW Translator validate the output.

When the generated document does not use these features, you should skip this page. It is valid for outbound documents only, not for acknowledgments.

The screenshot shows a dialog box titled "Partner Trade Agreement Options: EXAMPLE-X850". At the top, there is a dropdown menu for "Output" set to "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner", "Recipient Partner", and "Misc" (which is selected). The "Misc" tab contains several sections:

- Output Fields:** Two checkboxes, "Request Acknowledgment" and "Test", both of which are unchecked.
- Service Characters:** A grid of six text input fields for "Segment Terminator", "Component Delimiter", "Release Character", "Tag Delimiter", "Element Delimiter", and "Decimal Mark". Below this grid is a "Repeat Separator" input field.
- Checkboxes:** Three checkboxes are located at the bottom: "Document Level Break", "Acknowledgment Expected", and "Output Validation", all of which are unchecked.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Miscellaneous Page (Partner Trade Agreement Options Window)

Request Acknowledgment

A check in this box sets the internal field value **Acknowledgment Request** to true with a value of **1**. The map that generates the output wrapper must use this internal field to generate an element that represents a request acknowledgment flag for the standard. For acknowledgment maps that you create, you may need to change the internal TRM value to one that the standard requires using a cross-reference.

When the TRM generates an acknowledgment, the **Acknowledgment Request** internal field is set to false with a value of **0** (zero). The map for the acknowledgment wrapper moves this value to the appropriate element, because you should never request an acknowledgment for an acknowledgment itself.

Test

A check in this field sets the internal field value **Test Indicator** to true with a value of **1**. The map that generates the output wrapper must use this internal field to generate an element that represents a test flag for the standard. You may need to change the internal TRM value to one that the standard requires using a cross-reference.

When the TRM generates an acknowledgment, this flag is set to true with a value of **1** for the appropriate element in the acknowledgment wrapper when the value in the incoming document wrapper is set to true with a value of **1** or **T**. If the value in the incoming document was false (anything other than **1** or **T**), the acknowledgment wrapper map sets the appropriate element in the wrapper to false (**0** for an EDIFACT wrapper or **P** for an X12 wrapper).

Service Characters

These service characters represent the service characters that would be used to generate delimited data for outbound documents and override the service characters defined for the outbound wrapper.

Segment Terminator

Use this field to override the segment terminator character specified for the outbound wrapper. The segment terminator is the character used during generation to mark the end of segments. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Tag Delimiter

Use this field to override the tag delimiter character specified for the outbound wrapper. The tag delimiter is the character used during generation to mark the end of the segment tag. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the element delimiter. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

NOTE: Tag delimiters are not used by X12 or EDIFACT, but are used by TRADACOMS in Europe.

Element Delimiter

Use this field to override the element delimiter character specified for the outbound wrapper. The element delimiter is the character used during generation to mark the end of an element. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the tag delimiter. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Repetition Separator

Use this field to override the repetition separator character specified for the outbound wrapper. This value is used when there is a value or an offset for the Repetition Separator on Wrapper Properties, and when there is a repetition >1 for the composite or the element on the Segment window.

The repetition separator is the character used during parsing or generation to distinguish multiple occurrences of a composite or simple element within a segment. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Component Delimiter

Use this field to override the component delimiter character specified for the outbound wrapper. The component delimiter is the character used during generation to mark the end of a component element within a composite. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Release Character

Use this field to override the release character specified for the outbound wrapper. The release character is the character used during generation to mark the character that follows it as a true character, rather than one of these service characters, thus releasing the character from its normal duties as an element separator, for instance. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by `\x` followed by the number, such as, `\x13` to represent hex 13.

Decimal Mark

Use this field to override the decimal mark character specified for the outbound wrapper. The decimal mark is the character used during generation in numeric values that use an explicit decimal. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character. Hex numeric values must be preceded by `\x` followed by the number, such as, `\x13` to represent hex 13.

Document Level Break

Select this box to create an unconditional break with any previously generated documents that are in the current interchange. This document will be placed in its own interchange, and new interchanges will be created for subsequent documents as required.

Acknowledgment Expected

This allows you to specify whether an acknowledgment is expected in response to this generated document. The receipt of the expected acknowledgment can then be tracked using document reconciliation.

Output Validation

This allows you to validate the output to determine compliance with the definitions for the output wrappers and documents. Use this to assure that the output conforms to the output standard. MW Translator always validates the input, and this is the same process for the output. Be aware that output validation occurs during the generation phase of output. This means that any generation issues, such as element truncation, will cause validation to fail. Allowing elements to be truncated by simply using drag-and-drop is valid to create output, but if you check this flag, you must explicitly map the elements to avoid validation failure.

NOTE: This option does not validate XML output.

Partner Acknowledgment Options Window

You can specify alternative sender and recipient location IDs that would be used by an acknowledgment. You can also specify whether to use alternative service characters or those used in the input stream.

Procedures

These procedures apply to acknowledgments only, not to outbound documents.

To Specify Alternative Source or Destination Locations For an Acknowledgment For Partners

By default, the TRM uses the same locations for acknowledgment wrappers that belong to the sending and recipient partners identified in the incoming wrapper. You can specify alternative locations by entering them in the acknowledgment properties.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partners**.
- 2 In the right pane, double-click the partner for which you want to define alternative locations.
The Partner window appears.
- 3 Select the **Acknowledgments** tab.
- 4 Do one of the following, depending on whether the acknowledgment appears on the list:
 - If the acknowledgment appears on the list, select it, and then choose the **Options** button.
– or –
 - If the acknowledgment does not appear on the list, you must add it. For instructions, refer to the topic, *To Specify an Acknowledgment for a Partner* (on page 779).

The **Partner Acknowledgment Options** window appears.

- 5 In the **From** box, type the source location for the acknowledgment.
- 6 In the **To** box, type the destination location for the acknowledgment.
- 7 Select **OK** to complete the process.

To Specify Alternative Service Characters For an Acknowledgment For Partners

By default, the TRM uses the same service characters for acknowledgment wrappers that were used in the incoming wrapper. You can specify alternative service characters by entering them in the acknowledgment properties.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partners**.
- 2 In the right pane, double-click the partner for which you want to define alternative service characters. The Partner window appears.
- 3 Select the **Acknowledgments** tab.
- 4 Do one of the following, depending on whether the acknowledgment appears on the list:
 - If the acknowledgment appears on the list, select it, and then choose the **Options** button.
– or –
 - If the acknowledgment does not appear on the list, you must add it. For instructions, refer to the topic *To Specify an Acknowledgment for a Partner* (on page 779).

The **Partner Acknowledgment Options** window appears.

- 5 In the **Service Characters** box, clear the **Copy from input stream** check box.
- 6 Type the appropriate service characters that you will use in their respective boxes.
- 7 Select **OK** to complete the process.

(Partner Acknowledgment Options) General Page

The **General** page of the Partner Acknowledgment Options window specifies those values that are alternatives to the default values for acknowledgments for source or destination locations and service characters. If you do not want to specify alternative acknowledgment information, you may leave this page blank.

The screenshot shows a dialog box titled "Partner Acknowledgment Options: EXAMPLE-X997" with a "General" tab selected. The dialog is divided into two main sections: "Send Outputs:" and "Service Characters".

Send Outputs:

- From: [Text Input Field]
- To: [Text Input Field]

Service Characters:

- Copy from input stream
- Segment Terminator: [Text Input Field containing "~"]
- Component Delimiter: [Text Input Field containing ":"]
- Release Character: [Text Input Field]
- Tag Delimiter: [Text Input Field containing "x"]
- Element Delimiter: [Text Input Field containing "x"]
- Decimal Mark: [Text Input Field]
- Repeat Separator: [Text Input Field]

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

General Page (Partner Acknowledgment Options Window)

Send Outputs

This specifies the source (from) and destination (to) locations to be used for the acknowledgment instead of the locations already identified. By default, the locations for an acknowledgment are derived from the definitions used to parse the inbound document to which the acknowledgment responds. The information for the sender and recipient of the inbound document is reversed for the responding acknowledgment. The location for the sender of the inbound document becomes the location for the recipient of the acknowledgment. The location for the recipient of the inbound document becomes the location for the sender of the acknowledgment. These fields allow you to override that default behavior. For more information about the default behavior, refer to the topic, *Step 19. Determine Routing for Backward Acknowledgments* (on page 103).

Copy from input stream

When the **Copy from input stream** box is checked, the service characters from the input wrapper will be used to generate the acknowledgment. When the box is cleared, you can specify alternative service characters for the acknowledgment. They override whatever delimiters are associated with the incoming wrapper.

Segment Terminator

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The segment terminator is the character used during generation to mark the end of segments. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Tag Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The tag delimiter is the character used during generation to mark the end of the segment tag. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the element delimiter. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

NOTE: Tag delimiters are not used by X12 or EDIFACT, but are used by TRADACOMS in Europe.

Element Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The element delimiter is the character used during generation to mark the end of an element. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the tag delimiter. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Repetition Separator

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. This value is used when there is a value or an offset for the Repetition Separator on Wrapper Properties, and when there is a repetition greater than one for the composite or the element on the Segment window.

The repetition separator is the character used during parsing or generation to distinguish multiple occurrences of a composite or simple element within a segment. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Component Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The component delimiter is the character used during generation to mark the end of a component element within a composite. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Release Character

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The release character is the character used during generation to mark the character that follows it as a true character, rather than one of these service characters, thus releasing the character from its normal duties as an element separator, for instance. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Decimal Mark

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The decimal mark is the character used during generation in numeric values that use an explicit decimal. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Partner Relationship Window

The Partner Relationship window allows you to define a partnership where the sender and receiver are both known and have some unique configuration that cannot be defined for the sending and recipient partners separately. For example, if the sending partner usually does not receive acknowledgments, but must do so with this recipient partner, you would create a partner relationship. Alternatively, if there is a special map that must be used for this sender and recipient that would be inappropriate for other senders, you would create a partner relationship.

You can select existing partners or create new ones for your relationship using the Partner Relationship window.

Procedures

Here are some procedures you can perform from the Partner Relationship window.

To Add a Partner Relationship

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partner Relationships**.
- 2 From the **File** menu, select **Add**.
The **Add Partner Relationship** dialog box appears.
- 3 In the **Partner Relationship Name** box, type a name for this partner relationship.
- 4 From the **Sender Name** list, select a partner definition that will represent the sender, or type the name of a new partner.
- 5 From the **Recipient Name** list, select a partner definition that will represent the recipient, or type the name of a new partner.
- 6 Select **OK**.

The partnership appears in the **Partner Relationships** folder.

The two partners are also listed in the **Partners** folder.

- 7 To complete the Partner Relationship information, refer to the instructions, *To Complete a Partner Relationship* (on page 824).

To Complete a Partner Relationship

This procedure will also add new partners in the **Partners** folder. Make sure the Partner Explorer window is *open* (on page 777).

- 1 From the left pane, select **Partner Relationships**.
- 2 From the right pane, select your new partner relationship.
The Partner Relationship window appears.
- 3 Select the **Sending Partner** tab, and type the IDs required to match this sending partner.
- 4 Select the **Recipient Partner** tab, and type the IDs required to match this recipient partner.
- 5 Select the **Trade Agreements** tab, and add a trade agreement profile to the list. For instructions refer to the topic, *To Specify a Trade Agreement Profile for a Partner Relationship* (on page 825).
- 6 If you want to return a backward acknowledgment to the sending partner, refer to the instructions, *To Specify an Acknowledgment for a Partner Relationship* (on page 824).
- 7 Select **OK**.

To Delete a Partner Relationship

This procedure deletes only the partner relationship in the **Partner Relationships** folder. It does not delete the individual partners in the **Partners** folder.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partner Relationships**.
- 2 In the right pane, select the partner relationship you want to delete.
- 3 From the **File** menu, select **Delete**.
The **Confirm** dialog box appears.
- 4 Select **Yes**.

To Specify an Acknowledgment for a Partner Relationship

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partner Relationships**.
- 2 In the right pane, double-click the partner relationship for which you want to specify an acknowledgment.
The Partner Relationship window appears.
- 3 Select the **Acknowledgments** tab.
- 4 Select the **Add** button.

The **Select Acknowledgment Profile** dialog box appears.

- 5 From the list of defined acknowledgment profiles, select the acknowledgment you want to return to the sending partner.

- 6 Select **OK**.

The Partner Relationship Acknowledgment Options window appears.

- 7 To use a different location than that used for the recipient partner, type a destination location in the **To** box. You can also type a **From** address.

On the processing report the sending partner location will be identified as the source and the recipient partner location will be the destination.

- 8 For a delimited standard, to specify different service characters from those specified for the outbound wrapper, type the service characters in the appropriate boxes.

- 9 Select **OK** to add the acknowledgment to the partner's list.

- 10 Select **OK** to complete the changes to the Partner Relationship.

To Specify a Trade Agreement Profile for a Partner Relationship

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partner Relationships**.

- 2 In the right pane, double-click the partner relationship for which you want to specify a trade agreement.

The Partner Relationship window appears.

- 3 Select the **Trade Agreements** tab.

- 4 Select **Add**.

The **Select Trade Agreement Profile** dialog box appears.

- 5 From the list of defined trade agreement profiles, select the profile you want to use to process documents received by the recipient partner.

- 6 Select **OK**.

The Partner Relationship Trade Agreement Options window appears.

- 7 If you do not want to use the same IDs for the output document that you specified for the input document when you defined the sending and recipient partners, do the following:

- a) Select the **Sending Partner** tab, and enter partner IDs and a location.
- b) Select the **Recipient Partner** tab, and enter partner IDs and a location.

On the processing report the sending partner location will be identified as the source and the recipient partner location will be the destination.

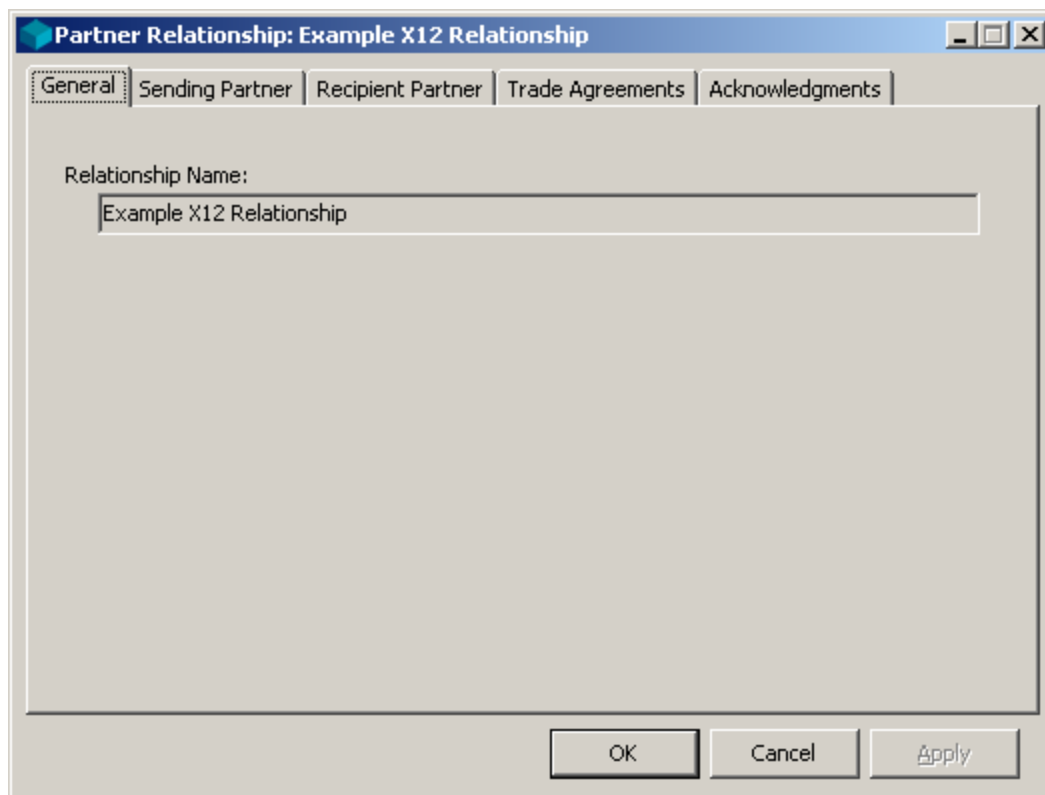
- 8 For a delimited standard, select the **Misc** tab to enter service characters to generate the data if they must be different from those specified for the outbound wrapper.

- 9 Select **OK** to add the trade agreement to the partner's list.

10 Select **OK** to complete the changes to the Partner Relationship.

(Partner Relationship) General Page

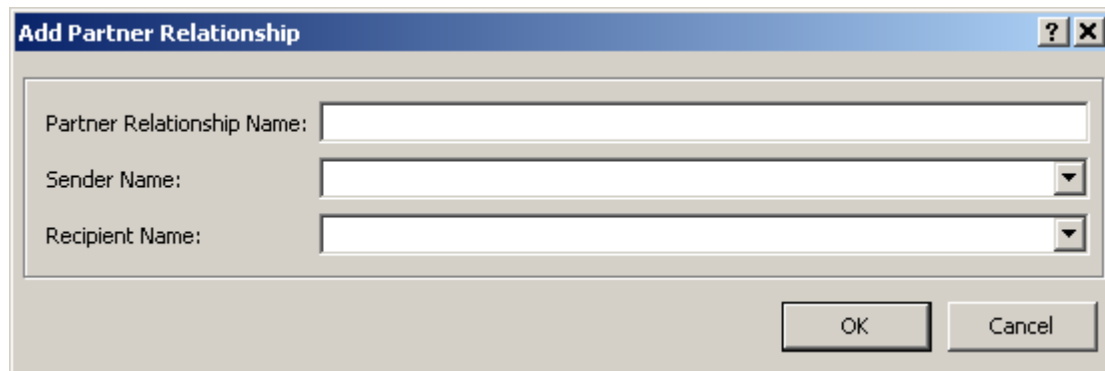
The **General** page displays the name you have given the relationship between sending and receiving partners. The relationship name is for display only. If you want to change the relationship name, you can do so, but you have to do it from Partner Explorer. You specify the name of the partner relationship in the **Add Partner Relationship** dialog box.



General Page (Partner Relationship Window)

Add Partner Relationship Dialog Box

This is a name that you give to the partnership between a specific sender and receiver. You enter the name when you *add* (on page 823) a partner relationship. You also select predefined partners as the sender and as the receiver.

The image shows a screenshot of a Windows-style dialog box titled "Add Partner Relationship". The dialog box has a blue title bar with a question mark icon and a close button (X). Inside the dialog, there are three input fields: "Partner Relationship Name:" followed by a text box, "Sender Name:" followed by a dropdown menu, and "Recipient Name:" followed by a dropdown menu. At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

Add Partner Relationship Dialog Box (Partner Relationship Window)

Partner Relationship Name

Enter a name for the partnership between a specific sender and receiver. It may be from 1 to 40 alphanumeric characters.

(Partner Relationship) Sending Partner Page

The **Sending Partner** page contains the partner name, location, and levels 1 (interchange) and 2 (functional group) of the 4 possible levels of IDs for the sending partner.

The screenshot shows a dialog box titled "Partner Relationship: Example X12 Relationship" with the "Sending Partner" tab selected. The dialog contains the following fields and controls:

- Sender Name:** A dropdown menu showing "Example X12 Sender".
- Location:** A text field containing "X12-SEND-LOC".
- Interchange:** A section containing:
 - ID:** A text field containing "ICH-SEND-ID".
 - Qual:** A text field containing "ZZ".
 - Int ID:** An empty text field.
 - Int ID2:** An empty text field.
- Functional Group:** A section containing:
 - ID:** A text field containing "FG-SEND-ID".
 - Qual:** An empty text field.
 - Int ID:** An empty text field.
 - Int ID2:** An empty text field.
- More...:** A button located below the Functional Group section.
- OK, Cancel, Apply:** Three buttons located at the bottom right of the dialog.

Sending Partner Page (Partner Relationship Window)

Sender Name

This partner definition appears in the **Partners** folder. To change the sending partner, select from among those on the defined list. To create a new partner, type the name.

Location (Sending Partner Page)

This is the location or location to which the TRM will route backward acknowledgments destined for this partner, unless routing is preempted by some other definition. For more information about how routing is determined for acknowledgments, refer to the topic, *Step 19. Determine Routing for Backward Acknowledgments* (on page 103).

ID (Sending Partner Page)

This field contains a value from one of the levels you can define for this partner. Refer to the Partner window field *ID (IDs Page)* (on page 789).

Qual (Sending Partner Page)

This field contains a value from one of the levels you can define for this partner. For more information, refer to the Partner window field, *Qual (IDs Page)* (on page 789).

Int ID (Sending Partner Page)

This field contains an additional ID value from one of the 4 possible levels you can define for this partner. For more information, refer to the Partner window field, *Int ID (IDs Page)* (on page 789).

Int ID2 (Sending Partner Page)

This field contains an additional ID value from one of the 4 possible levels you can define for this partner. For more information, refer to the Partner window field, *Int ID2 (IDs Page)* (on page 790).

More Button (Sending Partner Page, Partner Relationship)

If you want to specify IDs for wrapper levels 3 and 4 of the sending partner you choose the **More** button.

(Partner Relationship) Sending Partner (cont.) Page

The **Sending Partner** page contains the partner name, location, and levels 3 and 4 of the 4 possible levels of IDs for the sending partner.

The screenshot shows a software window titled "Partner Relationship: Example X12 Relationship". It features three main tabs: "Recipient Partner", "Trade Agreements", and "Acknowledgments". The "Sending Partner (cont.)" sub-tab is selected, showing two sections for "Level 3" and "Level 4". Each section contains four input fields: "ID", "Qual", "Int ID", and "Int ID2". A "Remove" button is positioned below the Level 3 section. At the bottom of the window are "OK", "Cancel", and "Apply" buttons.

Sending Partner (cont.) Page (Partner Relationship Window)

ID (Sending Partner (cont.) Page)

This field contains a value from one of the levels you can define for this partner. Refer to the Partner window field **ID (IDs Page)** (on page 789).

Qual (Sending Partner (cont.) Page)

This field contains a value from one of the levels you can define for this partner. For more information, refer to the Partner window field, **Qual (IDs Page)** (on page 789).

Int ID (Sending Partner (cont.) Page)

This field contains an additional ID value from one of the 4 possible levels you can define for this partner. For more information, refer to the Partner window field, *Int ID (IDs Page)* (on page 789).

Int ID2 (Sending Partner (cont.) Page)

This field contains an additional ID value from one of the 4 possible levels you can define for this partner. For more information, refer to the Partner window field, *Int ID2 (IDs Page)* (on page 790).

Remove Button

Choose the **Remove** button to remove ID values for wrapper levels 3 and 4.

(Partner Relationship) Recipient Partner Page

The **Recipient Partner** page contains the partner name, location, and levels 1 (interchange) and 2 (functional group) of the 4 levels of IDs for the recipient partner.

The screenshot shows a dialog box titled "Partner Relationship: Example X12 Relationship". It has five tabs: "General", "Sending Partner", "Recipient Partner", "Trade Agreements", and "Acknowledgments". The "Recipient Partner" tab is selected. The form contains the following fields:

- Recipient Name: Example X12 Recipient (dropdown menu)
- Location: X12-REC-LOC (text field)
- Interchange section:
 - ID: ICH-REC-ID (text field)
 - Qual: ZZ (text field)
 - Int ID: (empty text field)
 - Int ID2: (empty text field)
- Functional Group section:
 - ID: FG-REC-ID (text field)
 - Qual: (empty text field)
 - Int ID: (empty text field)
 - Int ID2: (empty text field)
- More... (button)
- OK, Cancel, and Apply (buttons)

Recipient Partner Page (Partner Relationship Window)

Recipient Name

This partner definition appears in the **Partners** folder. To change the recipient partner, select from among those on the defined list. To create a new partner, type the name.

Location (Recipient Partner Page)

This is the location or location to which the TRM will route output destined for this partner, unless routing is preempted by some other definition. For more information about how routing is determined for output, refer to the topic, *Step 14. Determine Routing and Miscellaneous Information (Translation)* (on page 91).

ID (Recipient Partner Page)

This field contains a value from one of the levels you can define for this partner. Refer to the Partner window field *ID (IDs Page)* (on page 789).

Qual (Recipient Partner Page)

This field contains a value from one of the levels you can define for this partner. For more information, refer to the Partner window field, *Qual (IDs Page)* (on page 789).

Int ID (Recipient Partner Page)

This field contains an additional ID value from one of the 4 possible levels you can define for this partner. For more information, refer to the Partner window field, *Int ID (IDs Page)* (on page 789).

Int ID2 (Recipient Partner Page)

This field contains an additional ID value from one of the 4 possible levels you can define for this partner. For more information, refer to the Partner window field, *Int ID2 (IDs Page)* (on page 790).

More Button (Recipient Partner Page, Partner Relationship)

Choose the **More** button to specify IDs for wrapper levels 3 and 4 of the recipient partner.

(Partner Relationship) Recipient Partner (cont.) Page

The **Recipient Partner (cont.)** page contains levels 3 and 4 of the 4 levels of IDs for the recipient partner.

The screenshot shows a software window titled "Partner Relationship: Example X12 Relationship". It features a tabbed interface with three main tabs: "General", "Sending Partner", and "Recipient Partner". The "Recipient Partner" tab is selected, and within it, the "Recipient Partner(cont.)" sub-tab is active. This sub-tab contains two sections, "Level 3" and "Level 4". Each section has four input fields: "ID", "Qual", "Int ID", and "Int ID2". A "Remove" button is positioned below the "Level 4" section. At the bottom of the window, there are three buttons: "OK", "Cancel", and "Apply".

Recipient Partner (cont.) Page (Partner Relationship Window)

ID (Recipient Partner Page)

This field contains a value from one of the levels you can define for this partner. Refer to the Partner window field **ID (IDs Page)** (on page 789).

Qual (Recipient Partner Page)

This field contains a value from one of the levels you can define for this partner. For more information, refer to the Partner window field, **Qual (IDs Page)** (on page 789).

Int ID (Recipient Partner Page)

This field contains an additional ID value from one of the 4 possible levels you can define for this partner. For more information, refer to the Partner window field, *Int ID (IDs Page)* (on page 789).

Int ID2 (Recipient Partner Page)

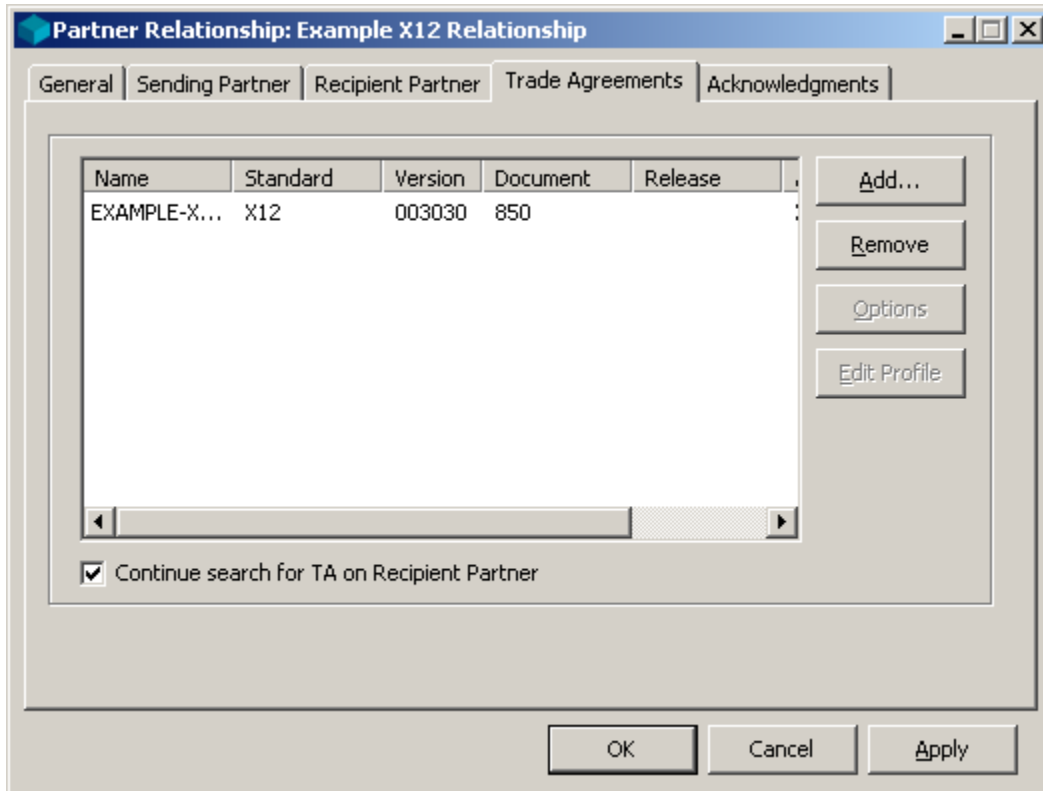
This field contains an additional ID value from one of the 4 possible levels you can define for this partner. For more information, refer to the Partner window field, *Int ID2 (IDs Page)* (on page 790).

Remove Button

Choose the **Remove** button to delete ID values for wrapper levels 3 and 4.

(Partner Relationship) Trade Agreements Page

The **Trade Agreements** page lists those trade agreements for this partnership. The TRM will go through the list attempting to match values it parsed from the input wrappers and stored in the internal locations against the values defined for the listed agreements.



Trade Agreements Page (Partner Relationships Window)

Name (Trade Agreements List)

The name uniquely identifies the trade agreement profile. It can contain 1 to 32 alphanumeric characters. It can have one or more output trade agreements defined for it, each of which can have multiple translation outputs based on different maps.

Standard (Trade Agreements List)

This is the standard of the incoming document. The value here must match the standard identified during the standard identification process.

Version (Trade Agreements List)

This is the interchange version of the standard identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Interchange Version**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Interchange Version**, then this internal field will be null, and will match the value in this version field only when it is blank.

Document (Trade Agreements List)

This is the document ID of the incoming document. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Document Id**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Document Id**, then this internal field will be null, and will match the value in this document field only when it is blank.

Release (Trade Agreements List)

This is the release of the standard identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Release**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Release**, then this internal field will be null, and will match the value in this Release field only when it is blank.

Agency (Trade Agreements List)

This is the agency identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Agency**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Agency**, then this internal field will be null and will match the value in this Agency field only when it is blank.

Assoc (Trade Agreements List)

This is the association identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Association Code**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Association Code**, then this internal field will be null, and will match the value in this Assoc field only when it is blank.

Continue Search for TA on Recipient Partner

This check box allows you to specify that you want to allow the TRM to search for other trade agreements that it can use if it none on the list match. This gives you explicit control over default processing. If you want the TRM to search this list and stop if there is no match, you should make sure this box is not checked. If you want the TRM to continue to search the trade agreement list on the recipient partner, check this box.

Add Button

The **Add** button allows you to add a new trade agreement to this partner relationship. It displays the **Select Trade Agreement Profile** dialog box.

Remove Button

The **Remove** button allows you to remove a trade agreement from this partner relationship.

Options Button

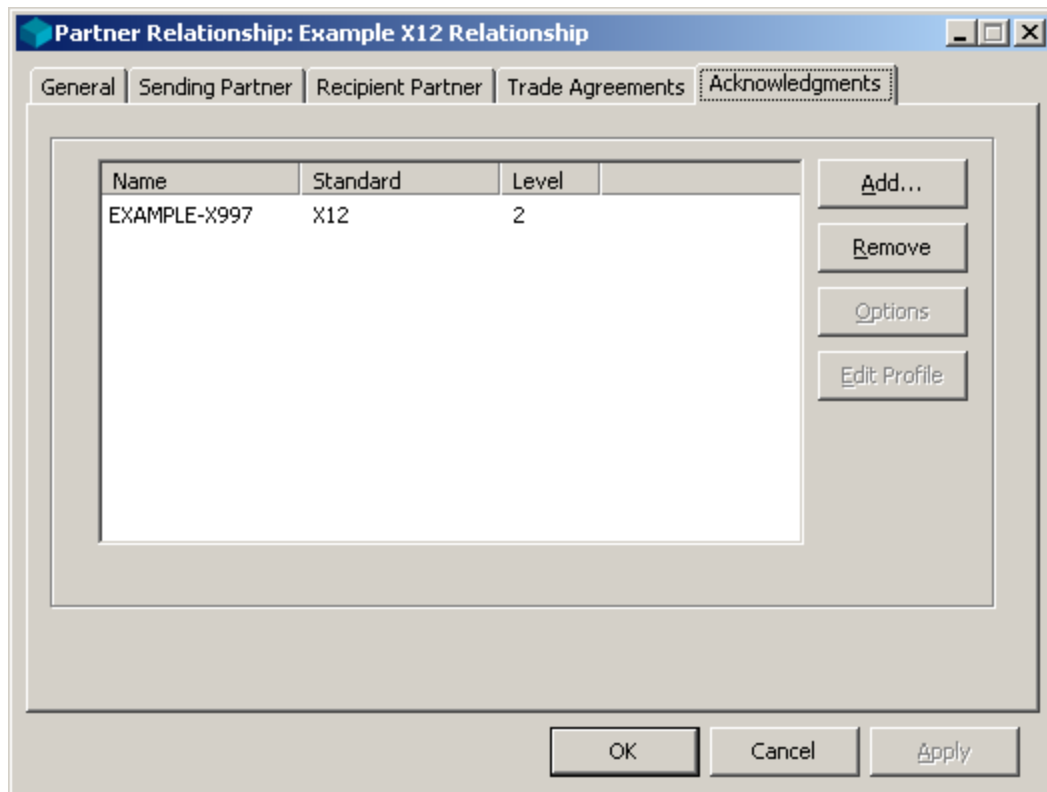
The **Options** button displays the **Partner Relationship Trade Agreement Options** window, which allows you to specify alternative IDs, location addresses, and service characters, for the output created using the selected trade agreement. You can only specify service characters when the output is a delimited standard. It also allows you to specify a value of true for test and acknowledgment-requested elements in the wrapper. In cases where the output sequences of the Trade Agreement profile, which is listed in the **Trade Agreements** folder of Data Explorer, do not match those of the Trade Agreement Options, the **Adjust Output Sequences** dialog box appears to allow you to fix the problem.

Edit Profile Button

The **Edit Profile** button allows you to change the definitions of the original trade agreement. This button is only active from the Workbench. For more information, refer to the description of the *Trade Agreement Profile window* (on page 652).

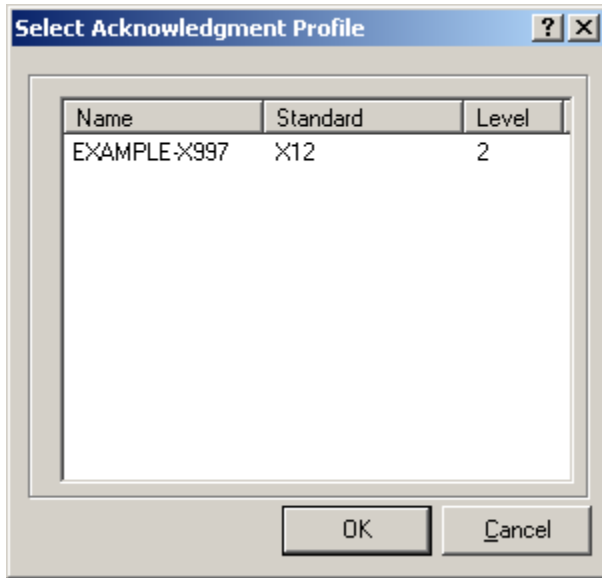
(Partner Relationship) Acknowledgments Page

The **Acknowledgments** page lists the acknowledgments that the sending partner will accept for this relationship.



Acknowledgments Page (Partner Relationship Window)

You can have multiple acknowledgment profiles for a given partner profile. This is where you identify which ones are associated with this partner relationship. You can add and remove defined acknowledgments for this partner list by using the **Add** and **Remove** buttons. You cannot change the profiles from here. When you select **Add**, you select an acknowledgment from the **Select Acknowledgment Profiles** dialog box.



Select Acknowledgment Profile Dialog Box (Acknowledgments Page)

Acknowledgments List

You can have multiple acknowledgment profiles for a given partner profile. This is where you identify which ones are associated with this partner relationship. You can add and remove defined acknowledgments for this partner list by using the **Add** and **Remove** buttons. You cannot change the profiles from here.

Name (Acknowledgments List)

The name uniquely identifies the Acknowledgment profile. This is where you identify which acknowledgments are to be returned to this partner when this partner is the sender.

Standard (Acknowledgments List)

To receive acknowledgments, you must specify a wrapper definition. The wrapper of the incoming document identified during standard identification must match the standard and version given here.

Level (Acknowledgments List)

The **Level** field identifies the level of wrapper to which the acknowledgment responds. It was assigned when the acknowledgment was defined.

There will be one acknowledgment for each occurrence of that level wrapper. The options are as follows:

Level	Description
0	Acknowledgment level not applicable. This is used for documents that do not receive acknowledgments themselves, and are therefore described as acknowledgments, but that in fact provide some other function, such as AUTACK for EDIFACT.
1	Interchange level acknowledgments are in response to all functional groups, if any, and all documents that the interchange contains. There will be one acknowledgment for each interchange level wrapper.
2	Functional group level acknowledgments are in response to all documents within a functional group. There will be one acknowledgment for each functional group level wrapper.

Add Button (Acknowledgments)

The **Add** button allows you to add a new acknowledgment to this partner relationship.

Remove Button (Acknowledgments)

The **Remove** button allows you to remove an acknowledgment from this partner relationship.

Options Button (Acknowledgments)

The **Options** button allows you to provide alternative location IDs for the sender and recipient. It also allows you to specify alternative service characters.

Edit Profile Button (Acknowledgments)

The **Edit Profile** button allows you to change the definitions of the original acknowledgment.

Partner Relationship Trade Agreement Options Window

This window allows you to define values that would be used as alternative values when the TRM generates the outbound document. By default, the TRM uses the same IDs for the outbound wrappers and the same location for the outbound documents that belong to the sending and recipient partners identified in the incoming wrapper. Since trade agreements can have multiple outputs, you can identify to which of the outputs this alternative information applies.

Output

Trade agreements can have multiple outputs. This selection box allows you to specify the output to which these overrides will apply. It also displays the sequential order of the output, if there are multiple from a given trade agreement.

Procedures

The procedures associated with trade agreement options do not affect backward acknowledgments.

To Specify Alternative Information in Outbound Wrappers for a Partner Relationship

By default, the TRM uses the same IDs and location for the outbound documents that belong to the sending and recipient partners identified in the incoming wrapper. You can specify alternative values in one of two ways: by using aliases or by specifying values directly. This procedure specifies values directly, and will override aliases if they already exist for a partner. This procedure does not affect backward acknowledgments.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partner Relationships**.
- 2 In the right pane, double-click the partner relationship for which you want to specify alternative partner IDs in the outbound wrapper or the destination location.

The Partner Relationship window appears.

- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending on whether the trade agreement appears on the list:
 - If the trade agreement appears on the list, select it and then choose the **Options** button.
The **Partner Relationship Trade Agreement Options** window appears.

– or –

- If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to the topic, *To Specify a Trade Agreement Profile for a Partner Relationship* (on page 825).
- 5 If you want to change the IDs or location for this specific partner relationship for the sender, recipient, or both, from the appropriate page, **Sending Partner** or **Recipient Partner**, do one of the following:
 - From the list, select an existing partner to use its ID(s) and location.
 - or –
 - Type a new partner name with new IDs and location. This creates a partner in the **Partners** folder.
 - 6 Choose **OK** to complete the process.

To Generate a Document in Its Own Interchange for a Partner Relationship

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partner Relationships**.
- 2 In the right pane, double-click the partner relationship on whose trade agreement you want to specify the unconditional break.

The Partner Relationship window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending on whether the trade agreement appears on the list:
 - If the trade agreement appears on the list, select it and then choose the **Options** button.

The **Partner Relationship Trade Agreement Options** window appears.

 - or –
 - If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to the topic, *To Specify a Trade Agreement Profile for a Partner Relationship* (on page 825).
- 5 Select the **Misc** tab, and then select the **Document Level Break** check box.

A check mark appears.
- 6 Select **OK**.

To Specify When a Document Requires an Acknowledgment for a Partner Relationship

You can keep track of documents that require responding acknowledgments using document reconciliation. To specify which documents expect acknowledgments, follow this procedure.

NOTE: You must also select **Enable Reconciliation** on the **MessageWay Settings** tab of the Operator program.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partner Relationships**.
- 2 In the right pane, double-click the partner relationship on whose trade agreement you want to specify that an acknowledgment is expected in response to this output.
The Partner Relationship window.
 - 3 Select the **Trade Agreements** tab.
 - 4 Do one of the following, depending on whether the trade agreement appears on the list:
 - If the trade agreement appears on the list, select it and then choose the **Options** button.
The **Partner Relationship Trade Agreement Options** window appears.
– or –
 - If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to the topic, *To Specify a Trade Agreement Profile for a Partner Relationship* (on page 825).
 - 5 Select the **Misc** tab, and then select the **Acknowledgment Expected** check box.
A check mark appears.
 - 6 Select **OK**.

(Partner Relationship Trade Agreement Options) Sending Partner Page

The **Sending Partner** page of the Partner Relationship Trade Agreement Options window specifies those values that will provide alternative partner IDs and source location for the sending partner for this output. You can have multiple outputs from one trade agreement, so you have the options of defining different values for each output. It is valid for outbound documents only, not for acknowledgments.

Partner Relationship Trade Agreement Options: EXAMPLE-X850

Output: 1,EXAMPLE,1,FPO 1 of 1

Sending Partner | Recipient Partner | Misc

Partner Name: Example Output Sending Partner

Location:

Override Location: TESTSEND-MAILBOX

Interchange

ID: TESTSEND Qual:

Int ID: Int ID2:

Functional Group

ID: Qual:

Int ID: Int ID2:

OK Cancel Apply

Sending Partner Page (Partner Relationship Trade Agreement Options Window)

Partner Name (Sending Partner)

This identifies a different partner as the sender, whose definitions will be used for the output wrapper. Use this to override the outbound sending partner. You can select a partner name from the existing definitions, or enter a new name, and add information to create a new partner.

Location (Sending Partner)

This is the source location defined for the sending partner.

Override Location

Enter a different location here to override the location listed for this partner.

ID (Sending Partner)

This field contains the partner ID for the outbound sending partner. For more information refer to the topic, *ID (IDs Page)* (on page 789).

Qual (Sending Partner)

This field contains the partner ID qualifier for the outbound sending partner.. For more information refer to the topic, *Qual (IDs Page)* (on page 789).

Int ID (Sending Partner)

This field contains the internal ID for the outbound sending partner. For more information refer to the topic, *Int ID (IDs Page)* (on page 789).

Int ID2 (Sending Partner)

This field contains the internal sub-ID for the outbound sending partner. For more information refer to the topic, *Int ID2 (IDs Page)* (on page 790).

(Partner Relationship Trade Agreement Options) Recipient Partner Page

This page allows you to specify alternative partner IDs and location for the recipient partner for this output. You can have multiple outputs from one trade agreement, so you have the options of defining different values for each output. It is valid for outbound documents only, not for acknowledgments. If you do not want to use alternative information for the recipient partner, you can leave this page blank.

The screenshot shows a dialog box titled "Partner Relationship Trade Agreement Options: EXAMPLE-X850". At the top, there is a dropdown menu for "Output" with the value "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner", "Recipient Partner" (which is selected), and "Misc". The "Recipient Partner" tab contains several input fields: "Partner Name" (a dropdown menu with "Example Output Recipient Partner" selected), "Location" (an empty text box), and "Override Location" (a text box containing "TESTREC-MAILBOX"). Below these are two sections: "Interchange" and "Functional Group". Each section has an "ID" field (with "TESTREC" in the Interchange ID field), a "Qual" field, and two "Int ID" fields. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Recipient Partner Page (Partner Relationship Trade Agreement Properties Window)

Partner Name (Recipient Partner)

This identifies a different partner as the recipient, whose definitions will be used for the output wrapper. Use this to override the outbound recipient partner. You can select a partner name from the existing definitions, or type a new name, and add information to create a new partner.

Location (Recipient Partner)

This is the location to which the TRM will route output destined for this recipient partner.

Override Location

Enter a different location here to override the location listed for this partner.

ID (Recipient Partner)

This field contains the partner ID for the outbound recipient partner. For more information refer to the topic, *ID (IDs Page)* (on page 789).

Qual (Recipient Partner)

This field contains the partner ID qualifier for the outbound recipient partner. For more information refer to the topic, *Qual (IDs Page)* (on page 789).

Int ID (Recipient Partner)

This field contains the internal ID for the outbound recipient partner. For more information refer to the topic, *Int ID (IDs Page)* (on page 789).

Int ID2 (Recipient Partner)

This field contains the internal sub-ID for the outbound recipient partner. For more information, refer to the topic, *Int ID2 (IDs Page)* (on page 790).

(Partner Relationship Trade Agreement Options) Misc Page

The **Miscellaneous** page allows you to specify the **Request Acknowledgment** and **Test** values in the outbound wrapper. You may also specify delimiters for the document that would override those specified for the outbound wrapper. You may also control document breaks in the output stream, specify that documents created with this trade agreement be marked to expect acknowledgments for reconciliation purposes and have MW Translator validate the output.

When the generated document does not use these features, you should skip this page. It is valid for outbound documents only, not for acknowledgments.

The screenshot shows a dialog box titled "Partner Relationship Trade Agreement Options: EXAMPLE-X850". At the top, there is a dropdown menu for "Output" with the value "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner", "Recipient Partner", and "Misc", with "Misc" being the active tab. The "Misc" page contains several sections of controls:

- Output Fields:** Two checkboxes, "Request Acknowledgment" and "Test", both of which are unchecked.
- Service Characters:** A group of six text input fields with corresponding labels: "Segment Terminator", "Component Delimiter", "Release Character", "Tag Delimiter", "Element Delimiter", and "Decimal Mark". Below these is a "Repeat Separator" field. All fields are currently empty.
- Document Level Break:** An unchecked checkbox.
- Acknowledgment Expected:** An unchecked checkbox.
- Output Validation:** An unchecked checkbox.

At the bottom of the dialog box are three buttons: "OK", "Cancel", and "Apply".

Miscellaneous Page (Partner Relationship Trade Agreement Options Window)

Request Acknowledgment

A check in this box sets the internal field value **Acknowledgment Request** to true, with a value of **1**. The map that generates the output wrapper must use this internal field to generate an element that represents a request acknowledgment flag for the standard. For acknowledgment maps that you create, you may need to change the internal TRM value to one that the standard requires using a cross-reference.

When the TRM generates an acknowledgment, the **Acknowledgment Request** internal field is set to false, with a value of **0** (zero). The map for the acknowledgment wrapper moves this value to the appropriate element, because you should never request an acknowledgment for an acknowledgment itself.

Test

A check in this field sets the internal field value **Test Indicator** to true using a value of **1**. The map that generates the output wrapper must use this internal field to generate an element that represents a test flag for the standard. You may need to change the internal TRM value to one that the standard requires using a cross-reference.

When the TRM generates an acknowledgment, this flag is set to true using a value of **1** for the appropriate element in the acknowledgment wrapper if the value in the incoming document wrapper was set to true (**1** or **T**). If the value in the incoming document was false (anything other than **1** or **T**), the acknowledgment wrapper map sets the appropriate element in the wrapper to false (**0** for an EDIFACT wrapper or **P** for an X12 wrapper).

Service Characters

These service characters represent the service characters that would be used to generate delimited data for outbound documents and override the service characters defined for the outbound wrapper.

Segment Terminator

Use this field to override the segment terminator character specified for the outbound wrapper. The segment terminator is the character used during generation to mark the end of segments. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Tag Delimiter

Use this field to override the tag delimiter character specified for the outbound wrapper. The tag delimiter is the character used during generation to mark the end of the segment tag. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the element delimiter. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

NOTE: Tag delimiters are not used by X12 or EDIFACT, but are used by TRADACOMS in Europe.

Element Delimiter

Use this field to override the element delimiter character specified for the outbound wrapper. The element delimiter is the character used during generation to mark the end of an element. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the tag delimiter. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Repetition Separator

Use this field to override the repetition separator character specified for the outbound wrapper. This value is used when there is a value or an offset for the Repetition Separator on Wrapper Properties, and when there is a repetition greater than one for the composite or the element on the Segment window.

The repetition separator is the character used during parsing or generation to distinguish multiple occurrences of a composite or simple element within a segment. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Component Delimiter

Use this field to override the component delimiter character specified for the outbound wrapper. The component delimiter is the character used during generation to mark the end of a component element within a composite. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Release Character

Use this field to override the release character specified for the outbound wrapper. The release character is the character used during generation to mark the character that follows it as a true character, rather than one of these service characters, thus releasing the character from its normal duties as an element separator, for instance. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Decimal Mark

Use this field to override the decimal mark character specified for the outbound wrapper. The decimal mark is the character used during generation in numeric values that use an explicit decimal. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Document Level Break

Select this box to create an unconditional break with any previously generated documents that are in the current interchange. This document will be placed in its own interchange, and new interchanges will be created for subsequent documents as required.

Acknowledgment Expected

This allows you to specify whether an acknowledgment is expected in response to this generated document. The receipt of the expected acknowledgment can then be tracked using document reconciliation.

Output Validation

This allows you to validate the output to determine compliance with the definitions for the output wrappers and documents. Use this to assure that the output conforms to the output standard. MW Translator always validates the input, and this is the same process for the output. Be aware that output validation occurs during the generation phase of output. This means that any generation issues, such as element truncation, will cause validation to fail. Allowing elements to be truncated by simply using drag-and-drop is valid to create output, but if you check this flag, you must explicitly map the elements to avoid validation failure.

NOTE: This option does not validate XML output.

Partner Relationship Acknowledgment Options Window

You can specify alternative sender and recipient location IDs to be used by an acknowledgment. You can also specify whether to use alternative service characters or those used in the input stream.

Procedures

This procedure applies to acknowledgments only, not to outbound documents.

To Specify Alternative Source or Destination Locations For an Acknowledgment For Partner Relationships

By default, the TRM uses the same locations for acknowledgment wrappers that belong to the sending and recipient partners identified in the incoming wrapper. You can specify alternative locations by entering them in the acknowledgment properties.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Partner Relationships**.
- 2 In the right pane, double-click the relationship for which you want to define alternative locations.
The Partner Relationship window.
- 3 Select the **Acknowledgments** tab.
- 4 Do one of the following, depending on whether the acknowledgment appears on the list:
 - If the acknowledgment appears on the list, select it, and then choose the **Options** button.
The **Acknowledgment Options** window appears.
– or –
 - If the acknowledgment does not appear on the list, you must add it. For instructions, refer to the topic, *To Specify an Acknowledgment for a Partner Relationship* (on page 824).
- 5 In the **From** box, type the source location for the acknowledgment.
- 6 In the **To** box, type the destination location for the acknowledgment.
- 7 Select the **OK** button to complete the process.

(Partner Relationship Acknowledgment Options) General Page

The **General** page of the Partner Relationship Acknowledgment Options window specifies those values that are alternatives to the default values for acknowledgments for source or destination locations and service characters. If you do not want to specify alternative acknowledgment information, you may leave this page blank.

The screenshot shows a dialog box titled "Partner Relationship Acknowledgment Options: EXAMPLE-X997". The "General" tab is selected. The dialog contains two main sections: "Send Outputs:" and "Service Characters".

Send Outputs:

- From:
- To:

Service Characters:

- Copy from input stream
- Segment Terminator
- Component Delimiter
- Release Character
- Tag Delimiter
- Element Delimiter
- Decimal Mark
- Repeat Separator

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

General Page (Partner Relationship Acknowledgment Options Window)

Send Outputs

This specifies the source (from) and destination (to) locations to be used for the acknowledgment instead of the locations already identified. By default, the locations for an acknowledgment are derived from the definitions used to parse the inbound document to which the acknowledgment responds. The information for the sender and recipient of the inbound document is reversed for the responding acknowledgment. The location for the sender of the inbound document becomes the location for the recipient of the acknowledgment. The location for the recipient of the inbound document becomes the location for the sender of the acknowledgment. These fields allow you to override that default behavior.

Copy from input stream

When the **Copy from input stream** box is checked, the service characters from the input wrapper will be used to generate the acknowledgment. When the box is cleared, you can specify alternative service characters for the acknowledgment. They override whatever delimiters are associated with the incoming wrapper.

Segment Terminator

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The segment terminator is the character used during generation to mark the end of segments. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Tag Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The tag delimiter is the character used during generation to mark the end of the segment tag. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the element delimiter. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

NOTE: Tag delimiters are not used by X12 or EDIFACT, but are used by TRADACOMS in Europe.

Element Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The element delimiter is the character used during generation to mark the end of an element. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the tag delimiter. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Repetition Separator

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. This value is used when there is a value or an offset for the Repetition Separator on Wrapper Properties, and when there is a repetition greater than one for the composite or the element on the Segment window, which are defined from the Workbench.

The repetition separator is the character used during parsing or generation to distinguish multiple occurrences of a composite or simple element within a segment. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Component Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The component delimiter is the character used during generation to mark the end of a component element within a composite. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Release Character

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The release character is the character used during generation to mark the character that follows it as a true character, rather than one of these service characters, thus releasing the character from its normal duties as an element separator, for instance. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Decimal Mark

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The decimal mark is the character used during generation in numeric values that use an explicit decimal. The field can contain any displayable ASCII character, or a hexadecimal value representing a non-displayable ASCII character. Hexadecimal numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hexadecimal 13.

Standard ID Window

The Standard ID window allows you to enter parameters to correctly identify the incoming wrapper standard to parse the wrapper data, and to identify alternative behavior for partners, routing the output, trade agreements, and acknowledgments.

Procedures

The following procedures are some tasks you can perform from the Standard ID window.

To Add a New Source Location for Standard Identification

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Locations/StdID**.
- 2 From the **File** menu, choose **Add**.
The **Enter Location Name** dialog box appears.
- 3 Type the name of a new location.
- 4 Select **OK**.
The Partner Explorer window appears.
- 5 Now, you must specify a wrapper to match against your incoming data. To proceed with this task, refer to the procedure, *To Add a Wrapper to a Source Location for Standard Identification* (on page 857).

To Add a Wrapper to a Source Location for Standard Identification

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, expand the **Locations/StdID** folder, and select the appropriate location.
- 2 From the **File** menu, select **Add**.
The **Select Inbound Wrapper** dialog box appears.

- 3 In the right pane from the display of available wrappers, select the wrapper that you want the TRM to use to match the incoming data for this location.
- 4 Select **OK**.
The Partner Explorer window appears.
- 5 If you want, you can specify additional properties to control the processing of the incoming data, following these procedures:
 - *To Add Matching Criteria for Standard Identification* (on page 858)
 - *To Specify Default Processing Parameters* (on page 859)

To Add Matching Criteria for Standard Identification

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, expand the **Locations/StdID** folder, and select the appropriate location.
- 2 In the right pane, double-click the wrapper to which you want to add matching criteria.
The Standard ID Window appears.
- 3 Select the **General** tab, and select the **New** button.
- 4 In the Operator field, select the type of operation to be performed on the incoming data from the list.
- 5 You can identify both an offset and a value (a) or a combination of segment, field, and subfield together with a value (b). You cannot combine an offset with segment, field or subfield. Follow the instructions for either a or b.
 - a) In the **Offset** field, enter the number of bytes from the beginning of the file where you want the search to begin.
 - b) In the **Segment** box, type the absolute sequence of the segment in the incoming data. Remember, you haven't identified the wrapper definition yet, so the TRM doesn't know what relative sequence number you might have given to the wrapper segment in the definition of the set. If you are identifying a segment tag, this field should be blank.

In the **Field** box, type the absolute sequence of the element within the segment. If you enter a value in the Segment field, you must enter a value here also.

In the **Sub Field** box, type the absolute sequence of the component within the composite, if required.
- 6 Type the characters against which you will compare the incoming data. These values are also matched for case. If you have embedded spaces, you must enclose the entire value in quotation marks.
- 7 Select **OK** to save your changes.

To Specify Default Processing Parameters

You can specify default processing by attaching the properties to the inbound wrapper identified in the Standard ID window. These defaults allow the TRM to continue processing when it cannot find appropriate information associated with the incoming partners. Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, expand the **Locations/StdID** folder, and select the appropriate location.
- 2 In the right pane, double-click the wrapper for which you want to specify default processing.
The Standard ID window appears.
- 3 Depending on the type of default processing you want to define for the wrapper, select the appropriate tab: **Partners, Routing, Trade Agreements, or Acknowledgments**.
- 4 Make changes as required.
- 5 Select **OK** to save your changes.

To Specify a Trade Agreement Profile as a Default on Standard ID

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, expand the **Locations/StdID** folder, and select the appropriate location.
- 2 In the right pane, double-click the standard for which you want to specify a trade agreement.
The Standard ID window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Select the **Add** button.
The **Select Trade Agreement Profile** dialog box appears.
- 5 To associate this trade agreement with a closed group, type the group name in the **Closed Group Name** box.
- 6 From the list of defined trade agreement profiles, select the profile you want to use.
- 7 Select the **OK** button to add the trade agreement to the Standard ID list.

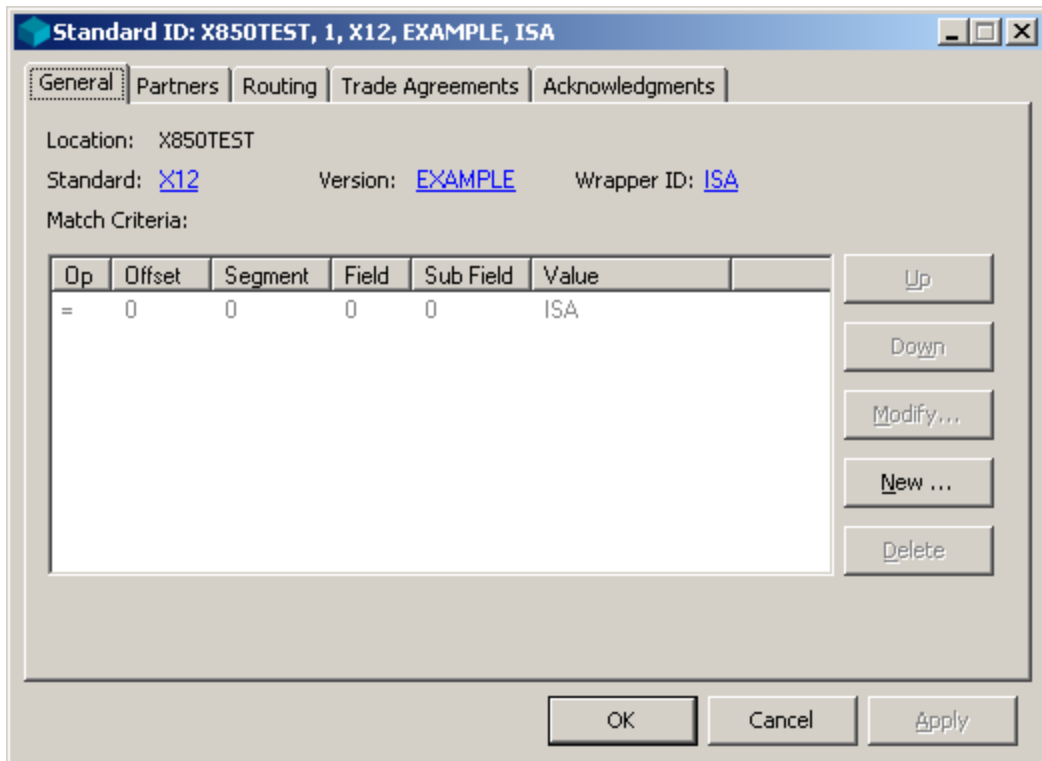
To Delete Matching Criteria for Standard Identification

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, expand the **Locations/StdID** folder, and select the appropriate location.
- 2 In the right pane, double-click the wrapper from which you want to delete matching criteria.
The Standard ID window appears.
- 3 Select the **General** tab, and select the criteria you want to delete in the detail area.
- 4 Select the **Delete** button.
- 5 Select **OK** to save your changes.

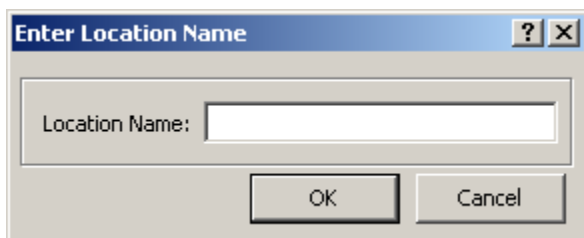
(Standard ID) General Page

The **General** page allows you to add, delete, and modify the criteria used to match the selected wrapper with the incoming data. The Workbench already contains the matching criteria for the public standard wrappers that it supports. When you load one of these standards, default matching criteria are also loaded as part of the Wrapper definition. You can see the criteria on the **StdID Match** tab of the Wrapper Properties window. You can modify all matching criteria, including those for public standards.



General Page (Standard ID Window)

The Standard ID window appears after you **add a source location** (on page 857), which displays the **Enter Location Name** dialog box.



Enter Location Name Dialog Box

Location Name

The **Enter Location Name** dialog box allows you to specify a name for the source location. Type a name from 1 to 128 characters long. This name then appears under the **Locations/StdID** folder in Partner Explorer.

The location name allows you to send input streams to the TRM to optimize the matching process. You can specify a new location for incoming data or use the **<Default>** location. The TRM uses the incoming location to determine where to look for a matching wrapper definition it should use to parse the incoming data. If it has no source location, it will attempt to match the criteria in the **<Default>** location.

Messaging systems, in particular, use location locations to determine document routing and services. The inbound location identifies different things depending on the platform you use for translation:

- on a Windows system using MessageWay, the inbound location is called the input location, which may be passed by an adapter or service that is capable of passing such information.
- on the PC platform using the Workbench, the inbound location comes from what you enter at the prompt when you run a test to process an input file.
- on the PC platform using CMC/MAPI, the inbound location comes from the sending e-mail address.



You must decide whether to create a new inbound location for an incoming standard or to use **<Default>**. If a standard is only received from one or a small number of locations, then enter those definitions within separate locations, otherwise use **<Default>**.

Standard Name

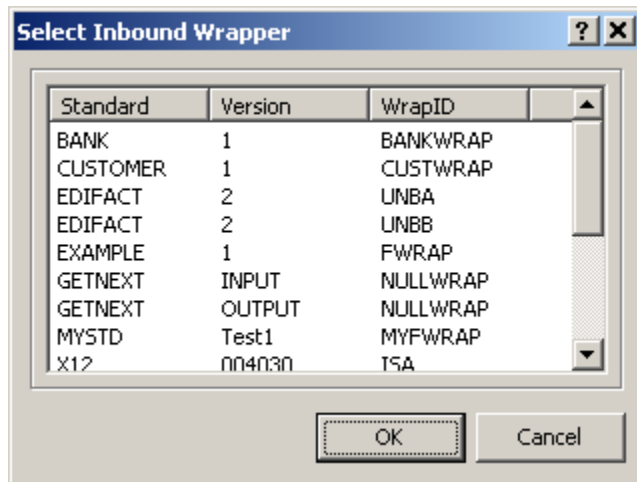
The standard name includes the standard, version, and wrapper ID for this wrapper. You must identify at least one wrapper definition for a standard version for a location. The TRM uses the wrapper definition to parse the incoming wrapper. It uses the parsed wrapper data to determine how to further process the information.

These values identify the wrapper definitions that are associated with the selected source location. Although you can have more than one wrapper associated with a particular location, we recommend that you do this only for the **<Default>** location, which typically identifies the wrappers for the public standards. If there is more than one wrapper defined for a particular source, the TRM will search the entries by order from the first entry at the top to the last entry at the bottom, using the first one that matches. For a given standard to match, all conditions in the **Matching Criteria** box must match. When there are no conditions for a particular standard, it will automatically match.

NOTE: Make sure that if you have a wrapper defined here without any matching data fields, that you place it at the end of the list, to be used as a default. Typically, you always want to include matching data fields for any wrapper definition that you enter, unless it is the only wrapper for the location.

Each entry occupies a position in the search order. If there is more than one wrapper per location, the TRM searches sequentially for a match, and chooses the first one that matches. You can change the order of sequencing by using the up  and down  arrows on the toolbar to move the selected item up or down in the list.

You select the standard name from a list of pre-defined wrappers in the **Select Inbound Wrapper** dialog box when you *add a wrapper for a standard version* (on page 857) to the location list.



Select Inbound Wrapper Dialog Box

Match Criteria

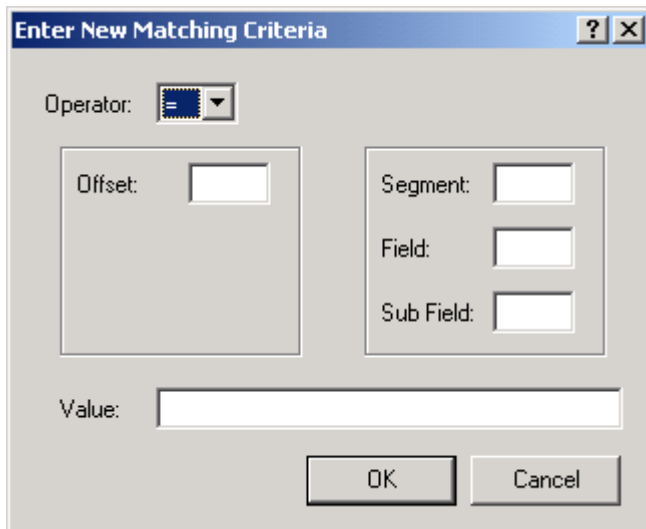
This list displays the information you enter to match a particular wrapper among several possibilities for the identified source location or for the **<Default>** location. If you selected a public standard wrapper, default matching criteria will appear automatically if they were specified as default values on the **StdID Match** tab of the Wrapper Properties window.

The matching value must occur within the first buffer of the data, typically 1k-4k, depending on your system. If there is only one wrapper associated with the location, there is no need to use the matching data fields.

IMPORTANT: To identify the incoming standard, the TRM reads the input file in raw mode as binary data. When it identifies the standard, it then re-opens the file. Then any required pre-processing user exit performs its tasks. Otherwise, when the IO Mode is Text, the TRM replaces line representations such as carriage-return/line-feed (CRLF), line-feed (LF) or record mark characters with an internal new-line (NL) character.

When there are multiple interchanges within the input file, subsequent standards must be identified with matching criteria that allow for a possible change in the number of characters when using offset values.

When you select the **Modify** or **New** button, the **Enter New Matching Criteria** dialog box appears.



The screenshot shows a dialog box titled "Enter New Matching Criteria". It features a title bar with a question mark icon and a close button. The main area contains an "Operator:" label followed by a dropdown menu with a blue icon. Below this are two columns of input fields: "Offset:" and "Segment:" on the left, and "Field:" and "Sub Field:" on the right. At the bottom, there is a "Value:" label followed by a text input field, and two buttons labeled "OK" and "Cancel".

Enter New Matching Criteria Dialog Box

Operator (Match Criteria)

You use one of these operators to compare the incoming value with the value given for this condition. The operator symbol identifies the type of condition that is applied, as follows:

Operator	Function
=	equal
<>	not equal
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal

Offset (Match Criteria)

The offset determines location from the beginning of the file where you begin counting. Zero (0) offset begins comparing with the first byte, which for delimited standards is the beginning of the segment tag. If you enter an offset, you cannot use segment, field, or sub-field.

Segment (Match Criteria)

The segment is the absolute sequence number of the segment that contains the incoming data for comparison. If you choose to match by segment, field, and potentially sub-field, you cannot enter an offset. If you enter a value for **Segment**, you must also enter a value for **Field**.

Field (Match Criteria)

The field corresponds to the element or composite element of the segment that contains the incoming data for comparison. It should contain the absolute number of the field, beginning with the first field, typically the segment tag, as field number 1. If you choose to match by segment, field, and potentially sub-field, you cannot enter an offset. If you enter a value for **Field**, you must also enter a value for **Segment**.

SubField (Match Criteria)

If the **Field** identifies a composite element, this identifies the component within the composite. The number of the first component element is field number 1. If you choose to match by segment, field, and potentially sub-field, you cannot enter an offset.

Value (Match Criteria)

This value is the string against which the input will be compared. Valid characters include any ASCII printable characters and hex values in the form, `\xnn`, where `nn` is the hex value, for example, `\x20` for a space. For EBCDIC data, you must place an **E** before the value, and the value must be enclosed in quotation marks, such as `E"HDR"`.

Trailing spaces entered for the value are deleted. To maintain trailing spaces to match an input value with trailing spaces, you may use a hex value at the end of the value field. For example, for an element of 15 characters, where the input value is **SENDPARTNER** followed by 4 spaces, in the value field of the **Enter New Matching Criteria** dialog box you would enter **SENDPARTNER** followed by three spaces and terminate with a hex space.

Up Button

The **Up** button allows you to move the selected criteria up one position in the matching order. The TRM terminates the matching process when it encounters a value that does not match the incoming data.

Down Button

The **Down** button allows you to move the selected criteria down one position in the matching order. The TRM terminates the matching process when it encounters a value that does not match the incoming data.

Modify Button

The **Modify** Button allows you to add matching criteria to match with the incoming data. This allows you to make distinctions between various definitions based on incoming wrapper data.

New Button

The **New** button allows you to enter new matching criteria.

Delete Button

The **Delete** button allows you to delete matching criteria.

(Standard ID) Partners Page

The **Partners** page within the Standard ID window allows you to enter rules about whether or not the TRM will use partner definitions to find a trade agreement. By checking the boxes in **Partner Definition Required** the TRM must match the IDs in the incoming wrapper to find a partner definition before additional processing occurs. This gives you strict control over how the TRM processes incoming data, because it does not allow default processing. In case no partner IDs or qualifiers exist in the incoming data for the sender or the recipient, and the **Partner Definition Required** boxes are not checked, this page also allows you to enter sending and recipient partners that can be used.

The screenshot shows a dialog box titled "Standard ID: X850TEST, 1, X12, EXAMPLE, ISA". The "Partners" tab is selected. The dialog contains the following fields and controls:

- Location: X850TEST
- Standard: X12
- Version: EXAMPLE
- Wrapper ID: ISA
- Partner Definition Required section with two checkboxes: Sending Partner and Recipient Partner.
- Default Sending Partner section with a "Name:" label and a dropdown menu.
- Default Recipient Partner section with a "Name:" label and a dropdown menu.
- Buttons at the bottom: OK, Cancel, and Apply.

Partners Page (Standard ID Window)

Sending Partner (Partner Definition Required)

This selection determines whether a partner must exist for the sender of the incoming document. The TRM uses the partner associated with the sender to determine if it needs to create a backward acknowledgment and if so, to find the sender's properties for acknowledgments. However, the TRM does not require partner records, since it can use the properties for the default sender. Checking the **Sending Partner** box ensures that the TRM uses the properties for the partner supplied in the incoming wrapper data. If this box is checked and the partner record does not exist, the process aborts early, saving processing time.

Recipient Partner (Partner Definition Required)

This selection determines whether a partner record must exist for the receiver of the incoming document. The TRM uses the partner associated with the receiver to find the partner, which determines how to process the output. The TRM also uses the partner profile for routing information. However, the TRM does not require partner profiles, since it can use the default receiver properties. Checking the **Recipient Partner** field ensures that the TRM uses the properties for the partner supplied in the incoming wrapper data. If this box is checked and the partner does not exist, the process aborts early, saving processing time.

Name (Default Sending Partner)

The default partners are only used when the inbound **ID** and **Qual** are both null in the internal table. If either the inbound **ID** or the inbound **Qual** is not null, then either the partner profile that matches these keys is used or *no* profile is used. You can select the name of the default sending partner from a list of valid partners.

Name (Default Recipient Partner)

The default partners are used only when the inbound **ID** and **Qual** are both null in the internal table. If either the inbound **ID** or the inbound **Qual** are not null, then either the partner profile that matches these keys is used or *no* profile is used. You select the name of the default recipient partner from a list of partners.

(Standard ID) Routing Page

The **Routing** page allows you to enter parameters to route output data or acknowledgments. This is important when you are using partner definitions if that information is lacking from locations defined for a partner relationship definition or its alternative trade agreement definitions, or from partner profiles or its alternative trade agreement definitions. If you are not using partners, the TRM must use the information on this page.

If you do provide routing information from the partner definitions, you can also override the source location of partner routing and use the original source location specified during standard identification.

The screenshot shows a dialog box titled "Standard ID: X850TEST, 1, X12, EXAMPLE, ISA". It has five tabs: "General", "Partners", "Routing" (which is selected), "Trade Agreements", and "Acknowledgments".

Under the "Routing" tab, the following information is displayed:

- Location: X850TEST
- Standard: X12
- Version: EXAMPLE
- Wrapper ID: ISA

There are two main sections for routing configuration:

- Default Output Routing:**
 - Data Field: Partner ID Source Address Use Source before Data Field
 - Partner Int ID Use Source before Partner Location
 - Default: Source:
 - Destination:
- Default Acknowledgment Routing:**
 - Data Field: Partner ID Source Address Use Source before Data Field
 - Partner Int ID Use Source before Partner Location
 - Default: Source:
 - Destination:

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Routing Page (Standard ID Window)

Default Output Routing

This series of check boxes provides a decision tree for the TRM to determine routing for the output data, in case there is no specific recipient partner information identified previously. The order of precedence is from left to right and then top to bottom, unless changed by the selection, **Use Source before Data Field**.

You can also override the source location of partner routing by selecting **Use Source before Partner Location**.

Data Field (Default Output Routing)

If the TRM has not found routing instructions for an output file, it will begin its default routing strategy with these values, unless you change the precedence by selecting the box **Use Source before Data Field**. If you select the check box **Data Field**, you can choose from one of the options, **Partner ID** or **Partner Int ID**. The TRM determines the partner ID by using the value in the element assigned to the internal fields **Rec Partner ID** or **Rec Partner Int ID**, respectively. You can find this assignment in the wrapper definition by viewing the segment definition online or by printing a wrapper report from Data Explorer.

Partner ID (Data Field, Default Output Routing)

If you select **Data Field** and **Partner ID**, then the TRM uses the location associated with the recipient partner ID. The TRM determines the partner ID by using the value in the element assigned to the internal field **Rec Partner ID**. You can find this assignment in the wrapper definition by viewing the segment definition online or by printing a wrapper report from Data Explorer.

Partner Int ID (Data Field, Default Output Routing)

If you select **Data Field** and **Partner Int ID**, then the TRM uses the location associated with the recipient partner internal ID. The TRM determines the partner internal ID by using the value in the element assigned to the internal field **Rec Partner Int ID**. You can find this assignment in the wrapper definition by viewing the segment definition online or by printing a wrapper report from Data Explorer.

Source Address (Default Output Routing)

When you check this box, the TRM will route the output to the source location, which is the location used during standard identification. If the **Data Field** box is also checked, it will take precedence. The other two **Use Source** options change the precedence.

Use Source before Data Field (Default Output Routing)

This check box allows you to reverse the normal precedence. If you also check the **Data Field** box, the TRM will use the **Source Address** before it will use the **Partner ID** or **Partner Int ID**.

Use Source before Partner Location (Default Output Routing)

This switch functions under different circumstances than any of the others on this page, although it also affects routing. The TRM uses the other options only if it does not find any other routing information during previous processing. The function of this switch is to let you override any partner-based routing that the TRM may already have found for the source location only. When you check this box, and if the TRM has already found routing instructions, it will replace the source with the name of the source location used during standard identification instead. This does not affect the routing for the destination location. This is the reverse of the default behavior of this field for acknowledgment routing

Default (Default Output Routing)

When you check **Default**, and if none of the other routing strategies have produced a routing address, the TRM will use the values here as routing addresses for the output data.

Source (Default Output Routing)

The source address will mean different things to different messaging systems. The **Source** might provide information that can affect routing, but the **Destination** box will provide the actual routing address.

Destination (Default Output Routing)

When you select **Default**, and if none of the other routing strategies has produced a routing address, the TRM will use the value in the **Destination** box to route the output data.

Default Acknowledgment Routing

This series of check boxes provides a decision tree to determine routing for the acknowledgments, in case there is no specific partner routing information identified previously. The order of precedence is from left to right and then top to bottom, unless changed by the selection, **Use Source before Data Field**. You can also override partner routing by selecting **Use Source before Partner Location**.

Data Field (Default Acknowledgment Routing)

If the TRM has not found routing instructions for an acknowledgment file, it will begin its default routing strategy with these values, unless you change the precedence by selecting the box **Use Source before Data Field**. When you select **Data Field**, you can choose from one of the two radio buttons, **Partner ID** or **Partner Int ID**. The TRM determines the ID by using the value in the element assigned to the internal fields **Send Partner ID** or **Send Partner Int ID**. You can find this assignment in the wrapper definition by viewing the segment definition online or by printing a wrapper report from Data Explorer.

Partner ID (Data Field, Default Acknowledgment Routing)

If you select **Partner ID**, then the TRM uses the location associated with the sending partner ID to route the acknowledgment. The TRM determines the sending partner ID by using the value in the element assigned to the internal field **Send Partner ID**. You can find this assignment in the wrapper definition by viewing the segment definition online or by printing a wrapper report from Data Explorer.

Partner Int ID (Data Field, Default Acknowledgment Routing)

When you select **Partner Int ID**, then the TRM uses the location associated with the sending partner internal ID to route the acknowledgment. The TRM determines the partner internal ID by using the value in the element assigned to the internal field **Send Partner Int ID**. You can find this assignment in the wrapper definition by viewing the segment definition online or by printing a wrapper report from Data Explorer.

Source Address (Default Acknowledgment Routing)

When you check this field, the TRM will route the acknowledgment to the source location, which is the location used during standard identification. The other two **Use Source** options change the precedence of what the TRM attempts to use.

Use Source before Data Field (Default Acknowledgment Routing)

This check box allows you to reverse the normal precedence. When you also check **Data Field** box, the TRM will use the **Source Address** before it will use the **Partner ID** or **Partner Int ID**.

Use Source before Partner Location (Default Acknowledgment Routing)

This switch functions under different circumstances than any of the others on this page, although it also affects routing. The TRM uses the other options only if it does not find any other routing information during previous processing. The function of this switch is to let you override any partner-based routing that the TRM may already have found for the destination. If you check this box and if the TRM has already found routing instructions, it will route the acknowledgment to the source location used during standard identification instead. In affect, it changes the destination location for the acknowledgment, but not the source location. This is the reverse of the default behavior of this field for output routing.

Default (Default Acknowledgment Routing)

When you select **Default**, and if none of the other routing strategies have produced a routing address, the TRM will use the values here as routing addresses for the acknowledgment.

Source (Default Acknowledgment Routing)

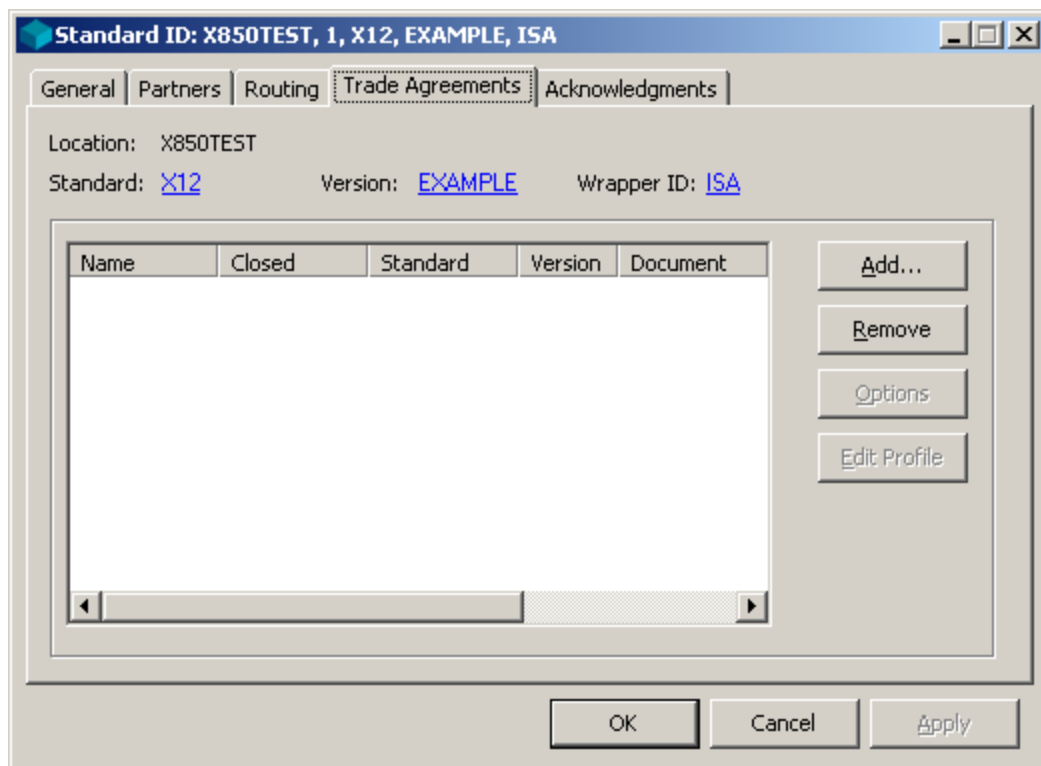
The source address will mean different things to different messaging systems. The **Source** might provide information that can affect routing, but the **Destination** box will provide the actual routing address.

Destination (Default Acknowledgment Routing)

When you select **Default**, and if none of the other routing strategies has produced a routing address, the TRM will use the value in the **Destination** box to route the acknowledgment.

(Standard ID) Trade Agreements Page

The TRM must use a trade agreement definition to continue processing the input data. The **Trade Agreements** page allows you to specify a list of trade agreements that supply a default. In case the TRM has not already found a trade agreement that was associated with a recipient partner definition, this list will allow it to make a final attempt to find one. The TRM will search the list for an appropriate match against the incoming data. If you are not using partner definitions that have associated trade agreements, the TRM must use a trade agreement definition from this page.

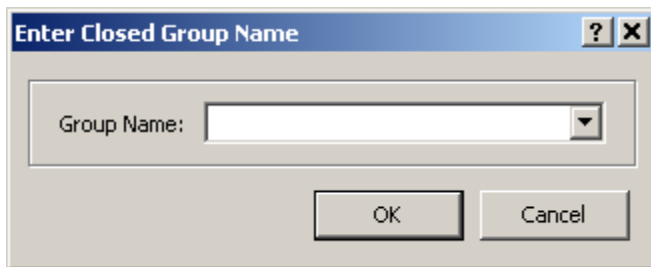


Trade Agreements Page (Standard ID Window)

Trade Agreements List

In order to process a document, the TRM must at least know the trade agreement so it can process the input. If the TRM cannot find a trade agreement based on a partner relationship or on a recipient partner definition, it tries to match one of the default trade agreements on the list. If there is no match or there are no trade agreements on the list, processing is terminated. At run time, the TRM will select the appropriate profile by matching the input data with the values that are listed here for the various columns.

If you want to assign a trade agreement already on the list to a group or change the assignment of a trade agreement to a group, you can right-click and select **Modify Group Name** from the menu. The **Enter Closed Group Name** dialog box appears.



Enter Closed Group Name Dialog Box

Name (Trade Agreements List)

The name uniquely identifies the trade agreement profile. It can contain 1 to 32 alphanumeric characters. It can have one or more output trade agreements defined for it, each of which can have multiple translation outputs based on different maps.

Closed (Trade Agreements List)

Enter a new group or select an existing group with which you want to associate this trade agreement. Trade agreements that are not associated with a group take precedence over those that are associated with a group. The TRM may select this trade agreement, assuming other fields also match, only if it has not already found a matching trade agreement on the list that belongs to no group. The group must match one of the groups listed on the **Groups** page of the sending partner's profile.

Standard (Trade Agreements List)

This is the standard of the incoming document. The value here must match the standard identified during the standard identification process.

Version (Trade Agreements List)

This is the interchange version of the standard identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Interchange Version**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Interchange Version**, then this internal field will be null, and will not match the value in this **Version** field.

Document (Trade Agreements List)

This is the document ID of the incoming document. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Document Id**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Document Id**, then this internal field will be null, and will not match the value in this **Document** field.

Release (Trade Agreements List)

This is the release of the standard identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Release**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Release**, then this internal field will be null, and will not match the value in this **Release** field.

Agency (Trade Agreements List)

This is the agency identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Agency**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Agency**, then this internal field will be null, and will not match the value in this **Agency** field.

Assoc (Trade Agreements List)

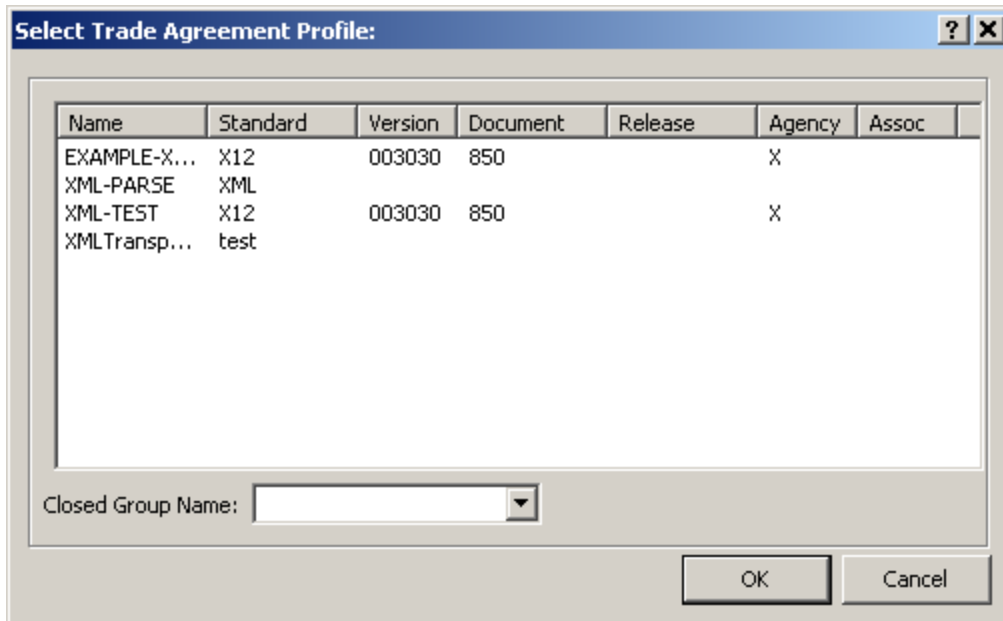
This is the association identified in the incoming wrapper set. If you enter a value here, it must match the value received in the element that is assigned to the internal field, **Association Code**. The element would have been assigned to this internal field when the segment was defined. If no element is assigned to **Association Code**, then this internal field will be null, and will not match the value in this **Assoc** field.

Add Button

To add trade agreements to this configuration, you select the **Add** button, and the **Select Trade Agreement Profile** dialog box appears.

Select Trade Agreement Profile

The **Select Trade Agreement Profile** dialog box allows you to choose from among existing trade agreement profile definitions by selecting a profile definition and then the **OK** button. You can also associate a trade agreement profile with a closed group. This dialog box appears when you select **Add** from the Trade Agreements tab of a Partner window, a Partner Relationship window or a Standard ID window.



Select Trade Agreement Profile Dialog Box

Remove Button

The **Remove** button allows you to remove a trade agreement profile from the list. This does not delete the profile, it only makes it unavailable as a possible choice for this input wrapper.

Options Button

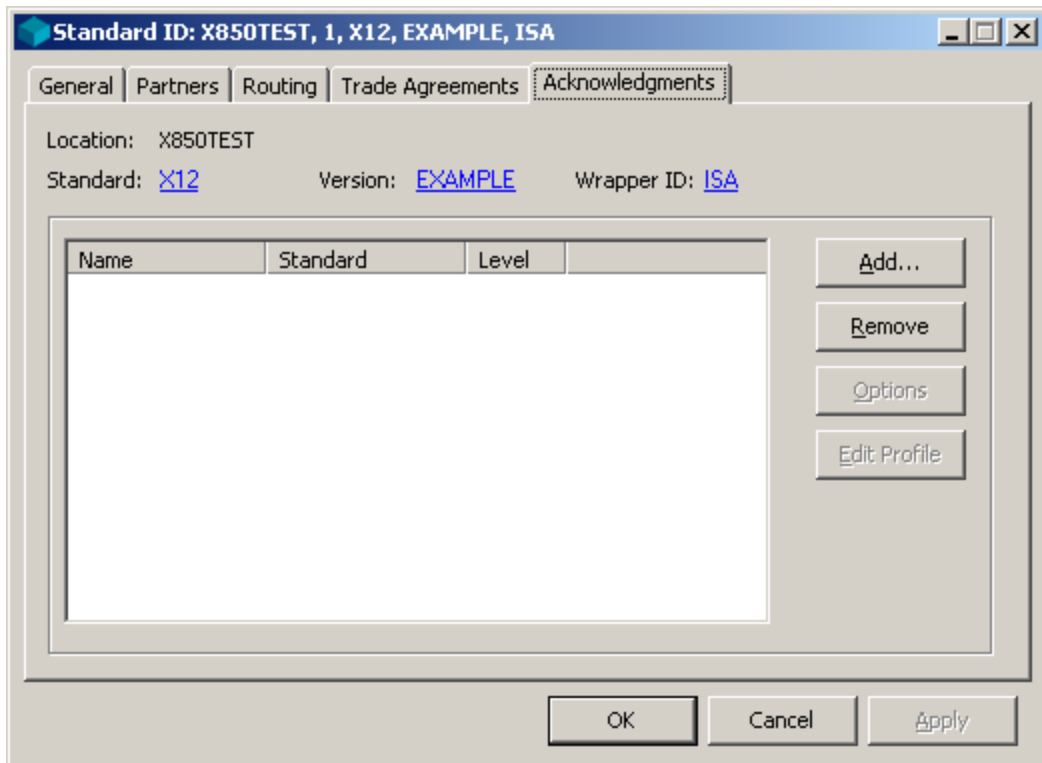
The **Options** button displays the **Standard ID Trade Agreement Options** window, which allows you to specify alternative IDs, location addresses, and service characters, for the output created using the selected trade agreement. You can only specify service characters when the output is a delimited standard. It also allows you to specify a value of true for test and acknowledgment-requested elements in the wrapper. In cases where the output sequences of the Trade Agreement profile, which is listed in the **Trade Agreements** folder of Data Explorer, do not match those of the Trade Agreement Options, the **Adjust Output Sequences** dialog box appears to allow you to fix the problem.

Edit Profile Button

The **Edit Profile** button is active only for the Workbench. It allows you to change the definition of the selected trade agreement profile. For more information, refer to the description of the *Trade Agreement Profile window* (on page 652).

(Standard ID) Acknowledgments Page

The TRM must use an acknowledgment definition to create an acknowledgment. The **Acknowledgments** page allows you to specify a list of acknowledgments that could supply a default. In case the TRM has not already found an acknowledgment that was associated with a partner relationship or on a sending partner definition, this list will allow it to make a final attempt to find one. The TRM will search the list for an appropriate match against the incoming data.



Acknowledgements Page (Standard ID Window)

Acknowledgments List

If the TRM cannot access the acknowledgment, it uses the default acknowledgment. If none is given, translation aborts. It needs the acknowledgment profile to generate acknowledgments that are returned to the sender.

Name (Acknowledgments List)

The name uniquely identifies the Acknowledgment profile. This is where you identify which acknowledgments are to be returned to this partner when this partner is the sender. You can add and remove defined trade agreements for this partner list by using the **Add** and **Remove** buttons. You cannot change the profiles from here.

Standard (Acknowledgments List)

To receive acknowledgments, you must specify a wrapper definition. The wrapper of the incoming document identified during standard identification must match the standard and version given here.

Level (Acknowledgments List)

The **Level** field identifies the level of wrapper to which the acknowledgment responds. It was assigned when the acknowledgment was defined.

There will be one acknowledgment for each occurrence of that level wrapper. The options are as follows:

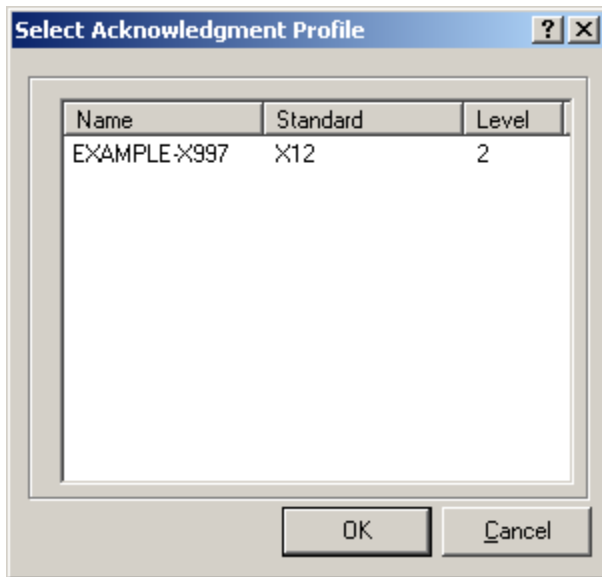
Level	Description
0	Acknowledgment level not applicable. This is used for documents that do not receive acknowledgments themselves, and are therefore described as acknowledgments, but that in fact provide some other function, such as AUTACK for EDIFACT.
1	Interchange level acknowledgments are in response to all functional groups, if any, and all documents that the interchange contains. There will be one acknowledgment for each interchange level wrapper.
2	Functional group level acknowledgments are in response to all documents within a functional group. There will be one acknowledgment for each functional group level wrapper.

Add Button (Standard ID Window, Acknowledgments Page)

You can add pre-defined acknowledgments to this list by selecting the **Add** button. This displays the **Select Acknowledgment Profile** dialog box.

Select Acknowledgment Profile Dialog Box (Standard ID Window, Acknowledgments Page)

You select a pre-defined acknowledgment from the **Select Acknowledgment Profiles** dialog box. The list displays information that should allow you to distinguish between profiles, such as the profile name, the standard to which it belongs, and the level of the acknowledgment. You can have multiple acknowledgment profiles. The TRM will generate any acknowledgments on the list that match the wrapper for the identified inbound standard that is the subject of the acknowledgment.



Select Acknowledgment Profile Dialog Box (Acknowledgments Page)

Remove Button (Standard ID Window, Acknowledgments Page)

The **Remove** button allows you to remove an acknowledgment profile from this list. It does not delete the profile.

Options Button (Standard ID Window, Acknowledgments Page)

The **Options** button allows you to enter alternative information for the trading partner in the outbound acknowledgment. This includes a different sender and/or receiver location IDs, and if you don't want to use the service characters that were used in the input data, you can specify different service characters here.

Edit Profile Button (Standard ID Window, Acknowledgments Page)

The **Edit Profile** button allows you to change the trade agreement profile information.

Standard ID Trade Agreement Options Window

You cannot create new trade agreement profiles from Partner Explorer. You can only create new trade agreement profiles from Data Explorer. You can, however, override the partner IDs and locations that would be used by a trade agreement from Partner Explorer. This page allows you to define values that will provide alternative values that would normally apply when the TRM generates the outbound document. By default, the TRM uses the same IDs for the outbound wrappers and the same location for the outbound documents that belong to the sending and recipient partners identified in the incoming wrapper. Since trade agreements can have multiple outputs, you can identify to which of the outputs this override information applies.

Output

Trade agreements can have multiple outputs. This selection box allows you to specify the output to which these overrides will apply. It also displays the sequential order of the output, if there are multiple from a given trade agreement.

Procedures

The procedures associated with trade agreement properties do not affect backward acknowledgments.

To Specify Alternative Information in Outbound Wrappers for a Standard ID Wrapper

As a default strategy, the TRM uses the same IDs and location for the outbound documents that belong to the sending and recipient partners identified in the incoming wrapper. You can specify alternative IDs and destination location in one of two ways: by using aliases or by specifying values directly. This procedure specifies values directly.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the appropriate location from the expanded view of the **Locations/StdID** folder.
- 2 In the right pane, double-click the Standard ID wrapper for which you want to specify alternative partner ID(s) in an outbound wrapper or an alternative destination location.

The Standard ID window appears.

- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending on whether the trade agreement appears on the list:

- If the trade agreement appears on the list, select it and then select the **Options** button.
The **Trade Agreement Options** window appears.
– or –
 - If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to *To Specify a Trade Agreement Profile as a Default on Standard ID* (on page 859).
- 5 Select the ID(s) and location of an existing partner from the drop-down list, or enter a new partner with new IDs and location.
 - 6 Select **OK**.

To Generate a Document in Its Own Interchange for a Standard ID Wrapper

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Locations/StdID**.
- 2 In the right pane, double-click the wrapper on whose trade agreement you want to specify the unconditional break.
The Standard ID window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending on whether the trade agreement appears on the list:
 - If the trade agreement appears on the list, select it and then select the **Options** button. The **Trade Agreement Options** window appears.
– or –
 - If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to *To Specify a Trade Agreement Profile as a Default on Standard ID* (on page 859).
- 5 Select the **Misc** tab.
- 6 Select the **Document Level Break** option.
- 7 Select **OK**.

To Specify When a Generated Document Requires an Acknowledgment for a Standard ID Wrapper

You can keep track of documents that require responding acknowledgments using document reconciliation. To specify which documents expect acknowledgments, follow this procedure.

NOTE: You must also select **Enable Reconciliation** on the **MessageWay Settings** page of the Operator program.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select **Locations/StdID**.
- 2 In the right pane, double-click the wrapper on whose trade agreement you want to specify that an acknowledgment is expected in response to this output.
The Standard ID window appears.
- 3 Select the **Trade Agreements** tab.
- 4 Do one of the following, depending whether the trade agreement appears on the list:
 - If the trade agreement appears on the list, select it and then select the **Options** button.
The **Trade Agreement Options** window appears.
– or –
 - If the trade agreement does not appear on the list, you must add it. For instructions to add a trade agreement, refer to *To Specify a Trade Agreement Profile as a Default on Standard ID* (on page 859).
- 5 Select the **Misc** tab.
- 6 Select the **Acknowledgment Expected** option.
- 7 Click **OK**.

(Standard ID Trade Agreement Options) Sending Partner Page

The **Sending Partner** page of the Standard ID Trade Agreement Options window specifies those values that will be used in place of the default values for the sending partner. If you do not want to specify alternative sending partner information, you can leave this page blank. It is valid for outbound documents only, not for acknowledgments.

The screenshot shows a dialog box titled "Standard ID Trade Agreement Options: EXAMPLE-X850". At the top, there is an "Output:" label followed by a dropdown menu containing "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner" (which is selected and highlighted with a dashed border), "Recipient Partner", and "Misc". The "Sending Partner" section contains several input fields: "Partner Name:" (a dropdown menu), "Location:" (a text box), and "Override Location:" (a text box). Below these are two sections: "Interchange" and "Functional Group". Each section has four input fields: "ID:", "Qual:", "Int ID:", and "Int ID2:". At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Sending Partner Page (Standard ID Trade Agreement Options Window)

Partner Name (Sending Partner)

This identifies a different partner as the sender, whose definitions will be used for the output wrapper. Use this to override the default outbound sending partner information. You can select a partner name from the existing definitions, or enter a new name, and add information to create a new partner.

Location (Sending Partner)

This is the source location defined for the sending partner.

Override Location

Enter a different location here to override the location listed for this partner.

ID (Sending Partner)

This field contains the partner ID for the outbound sending partner. For more information refer to, *ID (IDs Page)* (on page 789).

Qual (Sending Partner)

This field contains the partner ID qualifier for the outbound sending partner. For more information refer to, *Qual (IDs Page)* (on page 789).

Int ID (Sending Partner)

This field contains the internal ID for the outbound sending partner. For more information refer to, *Int ID (IDs Page)* (on page 789).

Int ID2 (Sending Partner)

This field contains the internal sub-ID for the outbound sending partner. For more information refer to, *Int ID2 (IDs Page)* (on page 790).

(Standard ID Trade Agreement Options) Recipient Partner Page

This page allows you to specify alternative partner IDs and location for the recipient partner. It is valid for outbound documents only, not for acknowledgments.

The screenshot shows a dialog box titled "Standard ID Trade Agreement Options: EXAMPLE-X850". At the top, there is an "Output:" field with a dropdown menu showing "1,EXAMPLE,1,FPO" and "1 of 1" to its right. Below this are three tabs: "Sending Partner", "Recipient Partner" (which is selected and highlighted with a dashed border), and "Misc". The "Recipient Partner" tab contains several input fields: "Partner Name:" with a dropdown arrow, "Location:" with a text box, and "Override Location:" with a text box. Below these are two sections: "Interchange" and "Functional Group". Each section has an "ID:" field with a text box and a "Qual:" field with a text box. Below each of these is an "Int ID:" field with a text box and an "Int ID2:" field with a text box. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Recipient Partner Page (Standard ID Trade Agreement Options)

Partner Name (Recipient Partner)

This identifies a different partner as the recipient, whose definitions will be used for the output wrapper. Use this to override the default outbound recipient partner information. You can select a partner name from the existing definitions, or enter a new name, and add information to create a new partner.

Location (Recipient Partner)

This is the location to which the TRM will route output destined for this recipient partner.

Override Location

Enter a different location here to override the one listed for this partner.

ID (Recipient Partner)

This field contains the partner ID for the outbound recipient partner. For more information refer to, *ID (IDs Page)* (on page 789).

Qual (Recipient Partner)

This field contains the partner ID qualifier for the outbound recipient partner. For more information refer to, *Qual (IDs Page)* (on page 789).

Int ID (Recipient Partner)

This field contains the internal ID for the outbound recipient partner. For more information refer to, *Int ID (IDs Page)* (on page 789).

Int ID2 (Recipient Partner)

This field contains the internal sub-ID for the outbound recipient partner. For more information refer to, *Int ID2 (IDs Page)* (on page 790).

(Standard ID Trade Agreement Options) Misc Page

The **Miscellaneous** page allows you to specify the **Request Acknowledgment** and **Test** values in the outbound wrapper. You may also specify delimiters for the document that would override those specified for the outbound wrapper. You may also control document breaks in the output stream, specify that documents created with this trade agreement be marked to expect acknowledgments for reconciliation purposes and have MW Translator validate the output.

When the generated document does not use these features, you should skip this page. It is valid for outbound documents only, not for acknowledgments.

The screenshot shows a dialog box titled "Standard ID Trade Agreement Options: EXAMPLE-X850". At the top, there is a dropdown menu for "Output:" with the value "1,EXAMPLE,1,FPO" and a "1 of 1" indicator. Below this are three tabs: "Sending Partner", "Recipient Partner", and "Misc", with "Misc" being the active tab. The "Misc" page contains several sections of controls:

- Output Fields:** Two checkboxes, "Request Acknowledgment" and "Test", both of which are currently unchecked.
- Service Characters:** A grid of six text input fields, each with a small square icon to its left. The fields are: "Segment Terminator", "Component Delimiter", "Release Character", "Tag Delimiter", "Element Delimiter", and "Repeat Separator".
- Document Level Break:** A checkbox, currently unchecked.
- Output Validation:** A checkbox, currently unchecked.
- Acknowledgment Expected:** A checkbox, currently unchecked.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

Miscellaneous Page (Standard ID Trade Agreement Options Window)

Request Acknowledgment

Check this box to set the internal field value **Acknowledgment Request** to true (**1**). The map that generates the output wrapper must use this internal field to generate an element that represents a request acknowledgment flag for the standard. For acknowledgment maps that you create, you may need to change the internal TRM value to one that the standard requires using a cross-reference.

When the TRM generates an acknowledgment, the **Acknowledgment Request** internal field is set to false (**0**). The map for the acknowledgment wrapper moves this value to the appropriate element, because you should never request an acknowledgment for an acknowledgment itself.

Test

A check in this field sets the internal field value **Test Indicator** to true (**1**). The map that generates the output wrapper must use this internal field to generate an element that represents a test flag for the standard. You may need to change the internal TRM value to one that the standard requires using a cross-reference.

When the TRM generates an acknowledgment, this flag is set to true (**1**) for the appropriate element in the acknowledgment wrapper if the value in the incoming document wrapper was set to true (**1** or **T**). If the value in the incoming document was false (anything other than **1** or **T**), the acknowledgment wrapper map sets the appropriate element in the wrapper to false (**0** for an EDIFACT wrapper or **P** for an X12 wrapper).

Service Characters

These service characters represent the service characters that would be used to generate delimited data for outbound documents and override the service characters defined for the outbound wrapper.

Segment Terminator

Use this field to override the segment terminator character specified for the outbound wrapper. The segment terminator is the character used during generation to mark the end of segments. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Tag Delimiter

Use this field to override the tag delimiter character specified for the outbound wrapper. The tag delimiter is the character used during generation to mark the end of the segment tag. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the element delimiter. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

NOTE: Tag delimiters are not used by X12 or EDIFACT, but are used by TRADACOMS in Europe.

Element Delimiter

Use this field to override the element delimiter character specified for the outbound wrapper. The element delimiter is the character used during generation to mark the end of an element. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the tag delimiter. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Repetition Separator

Use this field to override the repetition separator character specified for the outbound wrapper. This value is used when there is a value or an offset for the Repetition Separator on Wrapper Properties, and when there is a repetition >1 for the composite or the element on the Segment window.

The repetition separator is the character used during parsing or generation to distinguish multiple occurrences of a composite or simple element within a segment. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Component Delimiter

Use this field to override the component delimiter character specified for the outbound wrapper. The component delimiter is the character used during generation to mark the end of a component element within a composite. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Release Character

Use this field to override the release character specified for the outbound wrapper. The release character is the character used during generation to mark the character that follows it as a true character, rather than one of these service characters, thus releasing the character from its normal duties as an element separator, for instance. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by `\x` followed by the number, such as, `\x13` to represent hex 13.

Decimal Mark

Use this field to override the decimal mark character specified for the outbound wrapper. The decimal mark is the character used during generation in numeric values that use an explicit decimal. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character. Hex numeric values must be preceded by `\x` followed by the number, such as, `\x13` to represent hex 13.

Document Level Break

Select this box to create an unconditional break with any previously generated documents that are in the current interchange. This document will be placed in its own interchange, and new interchanges will be created for subsequent documents as required.

Acknowledgment Expected

This allows you to specify whether an acknowledgment is expected in response to this generated document. The receipt of the expected acknowledgment can then be tracked using document reconciliation.

Output Validation

This allows you to validate the output to determine compliance with the definitions for the output wrappers and documents. Use this to assure that the output conforms to the output standard. MW Translator always validates the input, and this is the same process for the output. Be aware that output validation occurs during the generation phase of output. This means that any generation issues, such as element truncation, will cause validation to fail. Allowing elements to be truncated by simply using drag-and-drop is valid to create output, but if you check this flag, you must explicitly map the elements to avoid validation failure.

NOTE: This option does not validate XML output.

Standard ID Acknowledgment Options Window

You can specify alternative source and destination locations to be used for an acknowledgment. You can also specify whether to use alternative service characters or those used in the input stream.

Procedures

These procedures apply to acknowledgments only, not to outbound documents.

To Specify Alternative Source or Destination Locations For an Acknowledgment For Standard ID

By default, the TRM uses the same locations for acknowledgment wrappers that belong to the sending and recipient partners identified in the incoming wrapper. You can specify alternative locations by entering them in the acknowledgment properties.

Make sure the Partner Explorer window is *open* (on page 777).

- 1 In the left pane, select the appropriate location from the **Locations/StdID** folder.
- 2 In the right pane, double-click the Standard ID wrapper for which you want to define alternative locations.
The Standard ID window appears.
- 3 Select the **Acknowledgments** tab.
- 4 Do one of the following depending whether the acknowledgment appears on the list:
 - If the acknowledgment appears on the list, select it and then select the **Options** button.
The **Acknowledgment Options** window appears.
– or –
 - If the acknowledgment does not appear on the list, you must add it.
- 5 In the **From** box, type the source location for the acknowledgment.
- 6 In the **To** box, type the destination location for the acknowledgment.
- 7 Select **OK**.

(Standard ID Acknowledgment Options) General Page

The **General** page of the Partner Acknowledgment Options window specifies those values that are alternatives to the default values for acknowledgments for source or destination locations and service characters. If you do not want to specify alternative acknowledgment information, you may leave this page blank.

The screenshot shows a dialog box titled "Standard ID Acknowledgment Options: EXAMPLE-X997" with a "General" tab selected. The dialog is divided into two main sections: "Send Outputs:" and "Service Characters".

Send Outputs:

- From:** A text input field.
- To:** A text input field.

Service Characters:

- Copy from input stream
- Segment Terminator:** A text input field containing "~".
- Component Delimiter:** A text input field containing ":".
- Release Character:** A text input field.
- Tag Delimiter:** A text input field containing "x".
- Element Delimiter:** A text input field containing "x".
- Decimal Mark:** A text input field.
- Repeat Separator:** A text input field.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

General Page (Standard ID Acknowledgment Options Window)

Send Outputs (Partner Acknowledgment Options)

This specifies the source (from) and destination (to) locations to be used for the acknowledgment instead of the locations already identified. By default, the locations for an acknowledgment are derived from the definitions used to parse the inbound document to which the acknowledgment responds. The information for the sender and recipient of the inbound document is reversed for the responding acknowledgment. The location for the sender of the inbound document becomes the location for the recipient of the acknowledgment. The location for the recipient of the inbound document becomes the location for the sender of the acknowledgment. These fields allow you to override that default behavior.

Copy from input stream

When the **Copy from input stream** box is checked, the service characters from the input wrapper will be used to generate the acknowledgment. When the box is cleared, you can specify alternative service characters for the acknowledgment. They override whatever delimiters are associated with the incoming wrapper.

Service Characters (Partner Acknowledgment Options)

If the **Copy from input stream** box is checked, the service characters will be dimmed. If the box is not checked, you can specify alternative service characters for the acknowledgment. These service characters will be used to generate delimited data. They override whatever delimiters are associated with the incoming wrapper.

Segment Terminator

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The segment terminator is the character used during generation to mark the end of segments. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Tag Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The tag delimiter is the character used during generation to mark the end of the segment tag. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the element delimiter. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

NOTE: Tag delimiters are not used by X12 or EDIFACT, but are used by TRADACOMS in Europe.

Element Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The element delimiter is the character used during generation to mark the end of an element. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character, except the tag delimiter. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Repetition Separator

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. This value is used when there is a value or an offset for the Repetition Separator on Wrapper Properties, and when there is a repetition >1 for the composite or the element on the Segment window, which are defined from the Workbench.

The repetition separator is the character used during parsing or generation to distinguish multiple occurrences of a composite or simple element within a segment. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Component Delimiter

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The component delimiter is the character used during generation to mark the end of a component element within a composite. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Release Character

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The release character is the character used during generation to mark the character that follows it as a true character, rather than one of these service characters, thus releasing the character from its normal duties as an element separator, for instance. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character, but it may not be the same as any other special character. Hex numeric values must be preceded by **\x** followed by the number, such as, **\x13** to represent hex 13.

Decimal Mark

Use this field to enter a value that differs from the wrapper of the input document to which this acknowledgment responds. The decimal mark is the character used during generation in numeric values that use an explicit decimal. The field can contain any displayable ASCII character, or a hex value representing a non-displayable ASCII character. Hex numeric values must be preceded by `\x` followed by the number, such as, `\x13` to represent hex 13.

Partner Wizard

Overview

You can use the Partner Wizard to enter new partner definitions. It will lead you through the steps to specify how you want to control your processing.

The wizard begins by asking you to select a trade agreement type:

- Specific Partner
- Open Trade
- Open Trade (without partners)
- Closed Group

Remember that the TRM must find a trade agreement in order to know how to process the information. You control when the trade agreement is used by associating it with partner relationship, partner or standard ID definitions. In doing so, you are in effect telling the TRM to use the trade agreement when:

- the sender and receiver IDs both match (Specific Partners/partner relationships)
- the receiver ID matches (Open Trade/partners)
- both the sender and the trade agreement belong to the same group (Closed Group)
- the incoming wrapper is identified (Open Trade).

The TRM searches these definitions in this order from most control (Specific Partners) to least control (Open Trade).

Although we offer you a choice among four trade agreement types, the possibilities are more than that. In essence, you can define your system to meet your specific requirements.

NOTE: You cannot use the Partner Wizard to maintain definitions. Once you have created definitions, you must use Partner Explorer to change or delete them.

Partner Wizard Window

The Partner Wizard window steps you through the process to define a new trading strategy, as stated on the **Welcome** page when you *open* (on page 898) the window. The Partner Wizard will define new definitions only.



A strategy is presented in two steps:

- 1 Specify how to control processing based on defined partners or default processing, and
- 2 Specify the inbound location and wrapper to identify the incoming standard needed to parse the wrapper data

To Open the Partner Wizard Window

- From the **Tools** menu, select **Partner Wizard**.

– or –

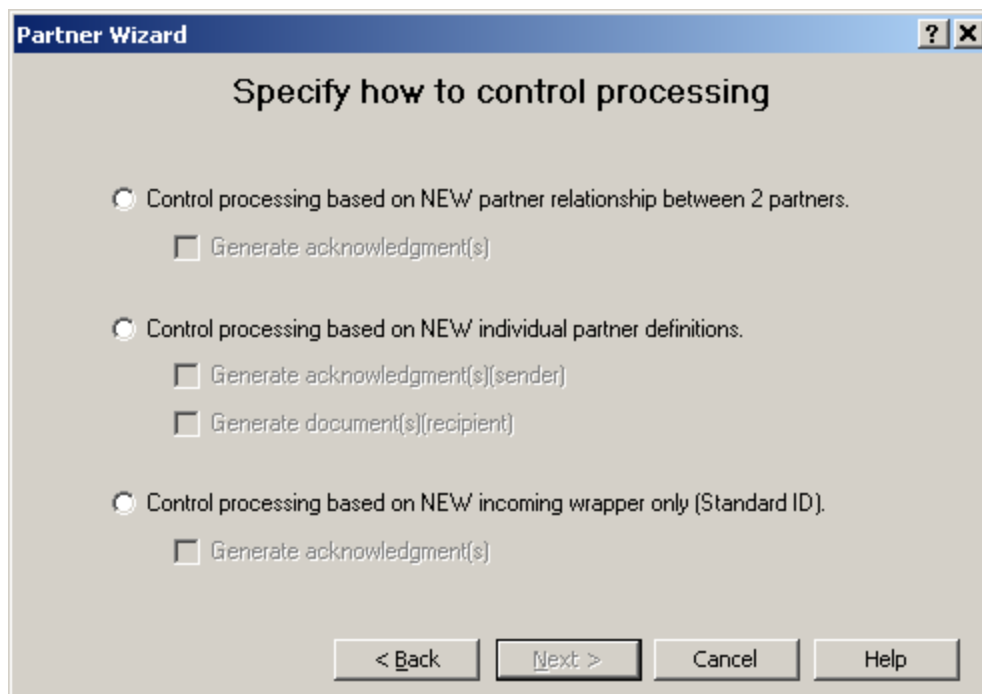
From the toolbar, select the **Partner Wizard** button



(Partner Wizard) Specify How to Control Processing Page

The purpose of this page is to determine with which definition you want to associate your trade agreement profile: partner relationship, recipient partner, or, as a default, standard ID/incoming wrapper. The trade agreement determines the work that must be done to generate output and specifies the definitions needed to complete the work.

This page also allows you to specify with which definition you want to associate an acknowledgment profile: partner relationship, recipient partner, or as a default, standard ID/incoming wrapper. The acknowledgment profile determines when an acknowledgment is to be generated and specifies the definitions needed to complete the work.



The screenshot shows a dialog box titled "Partner Wizard" with the subtitle "Specify how to control processing". It contains three radio button options, each with a corresponding checkbox:

- Control processing based on NEW partner relationship between 2 partners.
 - Generate acknowledgment(s)
- Control processing based on NEW individual partner definitions.
 - Generate acknowledgment(s)[sender]
 - Generate document(s)[recipient]
- Control processing based on NEW incoming wrapper only (Standard ID).
 - Generate acknowledgment(s)

At the bottom, there are four buttons: "< Back", "Next >", "Cancel", and "Help".

Specify How to Control Processing Page (Partner Wizard)

For a more thorough discussion about the difference between the options, refer to the chapters, *Defining Partners* (on page 243) and *Configuring Standard ID and Defaults* (on page 297).

Control Processing Based on NEW Partner Relationship

This option allows you to create a new partner relationship from existing partners or to create new partners for the relationship. You control processing by associating a trade agreement profile with this new partner relationship. The new relationship is listed in the **Partner Relationships** folder. This will also create a partner profile for the sender or receiver if either or both do not exist. You can find all partner definitions in the **Partners** folder. For more information, select the **Help** button.

A partner relationship is the most restrictive type of processing control that you can create. You would need to create a partner relationship if an individual partner definition would not suffice. For example, you would create a partner relationship rather than rely on the more general partner definition for:

- special handling of an acknowledgment: a sending partner typically received acknowledgments (acknowledgment defined on the partner's acknowledgment list), but the sending partner agreed to not receive an acknowledgment from a specific recipient (acknowledgment not defined on the partner relationship's acknowledgment list).
- special handling of input data: a recipient partner requires a special map to translate additional data that is not handled with the more generic map associated with the generic trade agreement for this type of input data.
- special user exits: the data from a particular sender requires special user exits that would not be required by the recipient partner's trade agreement for the same input data from other partners.

Generate Acknowledgment (Partner Relationship)

This option allows you to specify that an acknowledgment is required for this relationship. If this box is not checked, you will not be able to add an acknowledgment to this partner relationship using the wizard. You could always add one later from Partner Explorer.

Control Processing Based on NEW Individual Partner

This option allows you to create partner IDs that you can use to check against incoming data for security reasons. To specify trade agreements or acknowledgments for this partner you select the options separately. A partner is the most typical type of processing control, because it associates a rather generic trade agreement with a partner when it is the recipient, or a generic acknowledgment with the partner when it is the sender. Partners should use trade agreements or acknowledgments that satisfy most of their trading partners. If a partner requires something that cannot be satisfied with the generic processing, you would create a partner relationship for the two partners instead.

This option allows you to create a new partner by entering data, and possibly using an existing partner as a template. You control processing by associating one or more trade agreement profiles with this new partner. The new partner is listed in the **Partners** folder.

Generate Acknowledgment(s) (Partner)

This option allows you to specify that one or more acknowledgments are required for this partner. If this box is not checked, you will not be able to add an acknowledgment to this partner using the wizard. You could always add one later from Partner Explorer.

Generate Document(s) (Partner)

This option allows you to specify that one or more trade agreements are required for this partner. If this box is not checked, you will not be able to add a trade agreement to this partner using the wizard. You could always add one later from Partner Explorer.

Control Processing Based on NEW Incoming Wrapper

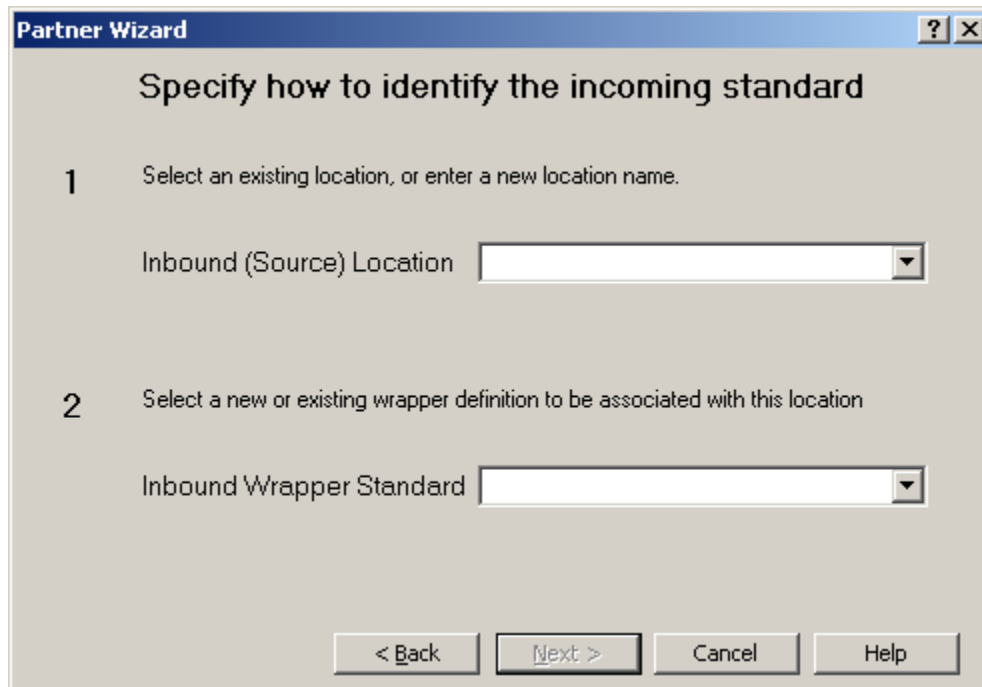
This option also allows you to identify a trade agreement profile without forcing a match of partner IDs for the receiver. The trade agreement is associated as a default with the standard version of the incoming wrapper, specified on the Standard ID window.

Generate Acknowledgment(s) (Standard ID/Wrapper)

This option allows you to specify that one or more acknowledgments are required for default processing. If this box is not checked, you will not be able to add an acknowledgment to this standard ID/wrapper using the wizard. You could always add one later from Partner Explorer.

(Partner Wizard) Specify How to Identify the Incoming Standard Page

The purpose of this page is to determine with which inbound location you want to associate the inbound wrapper definition: new location, existing location, or **<Default>** location. If you specify a new location, it is created in the **Locations/StdID** folder.



The screenshot shows a dialog box titled "Partner Wizard" with a close button (X) and a help button (?). The main title is "Specify how to identify the incoming standard".

Step 1: "Select an existing location, or enter a new location name." Below this is a label "Inbound (Source) Location" followed by a text input field with a drop-down arrow on the right.

Step 2: "Select a new or existing wrapper definition to be associated with this location." Below this is a label "Inbound Wrapper Standard" followed by a text input field with a drop-down arrow on the right.

At the bottom, there are four buttons: "< Back", "Next >", "Cancel", and "Help".

Specify How to Identify the Incoming Standard Page (Partner Wizard)

For a more thorough discussion about the difference between the options, refer to the chapter, *Configuring Standard ID and Defaults* (on page 297).

Inbound (Source) Location

This is where you specify your source location. The TRM uses this information during the standard identification process to help it sort through the identification sequence. You can select an existing location from the drop-down list including the **<Default>** location, or enter a new one. If the input location exists, it looks there only for a match with the incoming wrapper. If the inbound location does not exist or if there is no inbound location specified, the TRM looks in the **<Default>** location.

Inbound Wrapper Standard

If you have entered a new location, you must specify an inbound wrapper definition for that location to parse the data. For a new wrapper for the location, you will have the option to enter criteria that the TRM will use for matching. You may or may not want to enter matching criteria. If you enter no matching criteria, the TRM assumes a match, and uses this definition when it attempts to parse the data.

Back Button

Select the **Back** button to return to the previous definition page.

Next Button

Select the **Next** button to continue to the next definition page.

Cancel Button

Select the **Cancel** button to exit the Partner Wizard without saving any definitions.

(Partner Wizard) Match Criteria Page

You can enter information here to match a particular wrapper among several possibilities for the identified source location or to distinguish among multiple choices for the <Default> location. If there is only one wrapper associated with a location, there is no need to use the matching data fields, because a wrapper definition with no matching fields automatically matches. Matching criteria for public standard wrappers will be presented for you to accept or change as needed. For other standards, you must enter your own criteria.

If you later need to make changes, you must do so by accessing the matching criteria from the **General** page of the Standard ID window. For more information, refer to the topic, *Standard ID Window* (on page 857).

Partner Wizard

Match Criteria

Add or modify matching criteria which will be used at runtime to identify the selected wrapper when data is received from the selected location.

Location: X850TEST
Standard: X12 Version: 005010 Wrapper ID: ISA

Op	Offset	Segment	Field	Sub Field	Value
=	0	0	0	0	ISA
=	0	1	13	0	00501

Up
Down
New...
Modify...
Delete

< Back Next > Cancel Help

Match Criteria Page (Partner Wizard)

Match Criteria

This list displays the information you enter to match a particular wrapper among several possibilities for the identified source location or for the **<Default>** location. If you selected a public standard wrapper, there may already be matching criteria specified. If there is only one wrapper associated with the location, there is no need to use the matching data fields.

When you select the **Modify** or **New** button, the **Matching Criteria** dialog box appears.

Up Button

The **Up** button allows you to move the selected criteria up one position in the matching order. The TRM terminates the matching process when it encounters a value that does not match the incoming data.

Down Button

The **Down** button allows you to move the selected criteria down one position in the matching order. The TRM terminates the matching process when it encounters a value that does not match the incoming data.

Modify Button

The **Modify** Button allows you to add matching criteria that the TRM will use to match with the incoming data. This allows you to make distinctions between various definitions based on incoming wrapper data.

New Button

The **New** button allows you to enter new matching criteria.

Delete Button

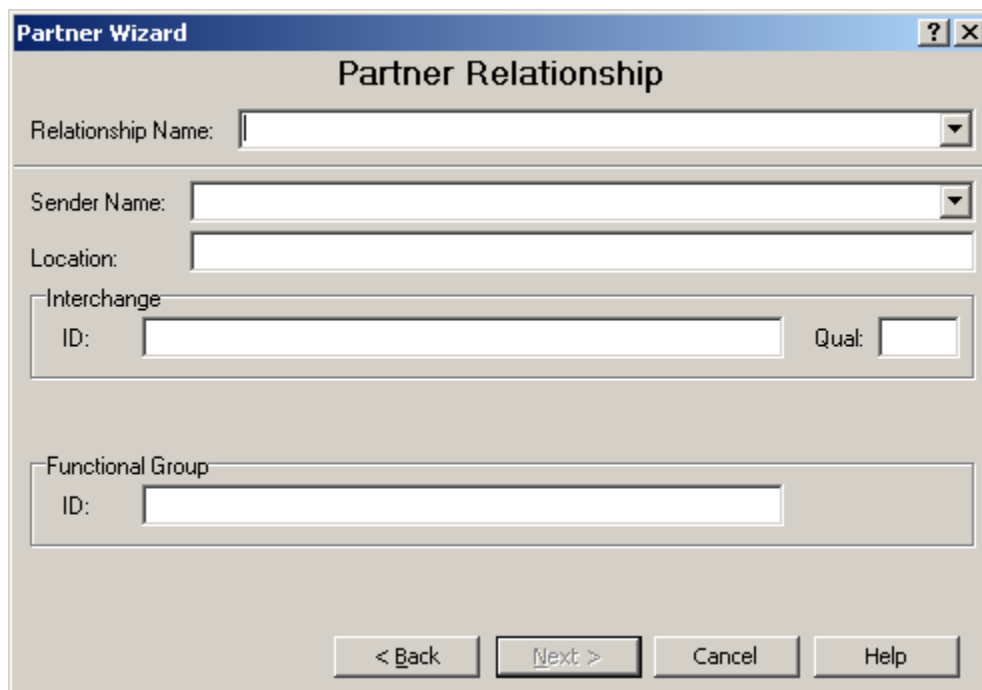
The **Delete** button allows you to delete matching criteria.

(Partner Wizard) Partner Relationship Page

This page allows you to enter information for the sending partner. If the sender does not exist, it will be created in the **Partners** folder. You also enter the name of a new relationship here. If you have selected a wrapper, the amount of information that you can enter on this page depends on the structure of the incoming wrapper you have selected.

NOTE: Using the Partner Wizard, you will only be able to enter a maximum of 2 levels of wrapper information. If you want to add information for levels 3 and 4, you will have to do so from Partner Explorer.

If you later need to make changes, you must do so by accessing the appropriate pages of the Partner Relationship window. For more information about this window, refer to the topic, *Partner Relationship Window* (on page 823).



The screenshot shows a dialog box titled "Partner Wizard" with a sub-header "Partner Relationship". The dialog contains several input fields and buttons:

- Relationship Name:** A text input field with a dropdown arrow on the right.
- Sender Name:** A text input field with a dropdown arrow on the right.
- Location:** A text input field.
- Interchange:** A section containing:
 - ID:** A text input field.
 - Qual:** A text input field.
- Functional Group:** A section containing:
 - ID:** A text input field.

At the bottom of the dialog are four buttons: "< Back", "Next >", "Cancel", and "Help".

Partner Relationship Page (Partner Wizard)

Relationship Name

This identifies a new and unique relationship. You can use the drop-down box to choose an existing relationship as a template. However, you must give the relationship a new name. This will be the name of the partner relationship that appears in the right pane of the Partner Explorer within the **Partner Relationships** folder.

Sender Name

This identifies the partner that will be the sender in this relationship. The pair of partners in this relationship must be unique for all the relationships that you have defined. If the sender does not exist, it will be created in the **Partners** folder.

Location

This is the location to which the TRM will route backward acknowledgments destined for this sending partner, unless routing is pre-empted by another setting.

Interchange ID and Qualifier

This group contains the interchange level ID and qualifier for this sending partner. If you have selected a pre-defined partner as the sender, you will not be able to change this information.

Functional Group ID

This group contains the functional group level ID for this sending partner. If you have selected a pre-defined partner as the sender, you will not be able to change this information. If you must add a qualifier, you should do so from Partner Explorer.

(Partner Wizard) Partner Relationship (cont.) Page

This page allows you to enter information for the recipient partner. If the partner does not exist, it will be created in the **Partners** folder. The relationship name is given in the header area.

If you have selected a wrapper, the amount of information that you can enter on this page depends on the structure of the incoming wrapper you have selected.

NOTE: Using the Partner Wizard, you will only be able to enter a maximum of 2 levels of wrapper information. If you want to add information for levels 3 and 4, you will have to do so from Partner Explorer.

If you later need to make changes, you must do so by accessing the appropriate pages of the Partner Relationship window. For more information about this window, refer to the topic, *Partner Relationship Window* (on page 823).

The screenshot shows a dialog box titled "Partner Wizard" with a sub-header "Partner Relationship (cont.)". The "Relationship Name" is "Example X12 Relationship22". The "Recipient Name" is a dropdown menu showing "Example X12 Recipient". The "Location" is "X12-REC-LOC". The "Interchange" section has an "ID" of "ICH-REC-ID" and a "Qual" of "ZZ". The "Functional Group" section has an "ID" of "FG-REC-ID". At the bottom are buttons for "< Back", "Next >", "Cancel", and "Help".

Partner Relationship (cont.) Page (Partner Wizard)

Recipient Name

This identifies the partner that will be the recipient in this relationship. The pair of partners in this relationship, sender and receiver, must be unique for all the relationships that you have defined. If the partner does not exist, it will be created in the **Partners** folder.

Location

This is the location to which the TRM will route output destined for this recipient partner, unless routing is pre-empted by some other definition.

Interchange ID and Qualifier

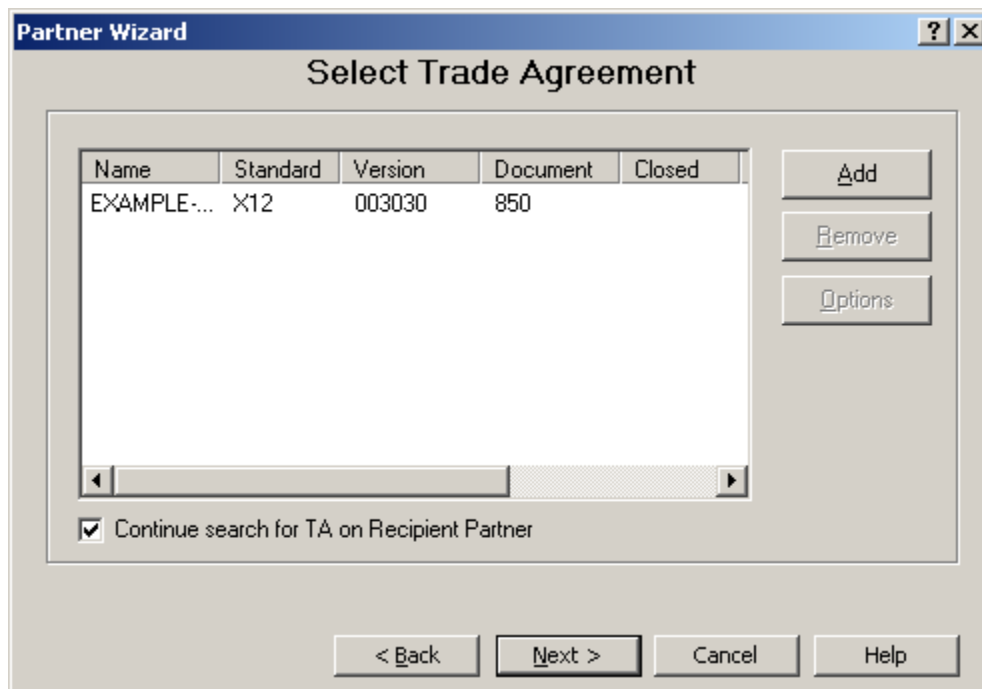
This group contains the interchange level ID and qualifier for this recipient partner. If you have selected a pre-defined partner as the recipient, you will not be able to change this information.

Functional Group ID

This group contains the functional group level ID for this recipient partner. If you have selected a pre-defined partner as the recipient, you will not be able to change this information. If you need to add a qualifier for this functional group ID, you must do so in Partner Explorer.

(Partner Wizard) Select Trade Agreement Page

The trade agreement is essential for any document processing. You associate the trade agreement with a partner relationship, a partner, or a standard ID definition. If you select **Continue Search for TA on**, you allow the TRM to continue searching for missing information, such as a trade agreement or a destination location for routing. For more information about how the TRM searches for a matching trade agreement, refer to the chapter, *Understanding Processing Flow* (on page 45).



Select Trade Agreement Page (Partner Wizard)

Trade Agreement List

The trade agreement list displays all trade agreements that this partner relationship, partner, or standard ID can use as a recipient.

Continue Search for TA on

This check box allows the TRM to search for other trade agreements that it might use if it finds none on this list or none on the list match. This gives you explicit control over default processing. If you want the TRM to search this list but not continue if there is no match, you should make sure this box is not checked. If you want the TRM to search the next trade agreement list, check this box.

(Partner Wizard) Partner Page

This page allows you enter a new partner profile. The **Partner** page appears when you have selected **Control Processing Based on Individual Partner** and **Generate Document(s)** or **Generate Acknowledgment(s)**. The information here represents the partner in a role as recipient, sender, or both. When you successfully complete the wizard process, this unique partner will be added to the **Partners** folder.

If you have selected a wrapper, the amount of information that you can enter on this page depends on the structure of the incoming wrapper you have selected.

NOTE: Using the Partner Wizard, you will only be able to enter a maximum of 2 levels of wrapper information. If you want to add information for levels 3 and 4, you will have to do so from Partner Explorer.

If you later need to make changes, you must do so by accessing the appropriate pages of the Partner window. For more information about this window, refer to the topic, **Partner Window** (on page 778).

The screenshot shows a dialog box titled "Partner Wizard" with a sub-header "Partner". The form contains the following fields:

- Partner Name: A dropdown menu.
- Location: A text input field.
- Interchange: A group box containing:
 - ID: A text input field.
 - Qual: A text input field.
- Functional Group: A group box containing:
 - ID: A text input field.

At the bottom of the dialog are four buttons: "< Back", "Next >", "Cancel", and "Help".

Partner Page (Partner Wizard)

Partner Name

This identifies the partner. You can select a pre-defined partner as a template, and give it a new name, changing any other information that is appropriate. The combination of values given in **ID**, **Qual**, and **Int ID** must be unique among all existing partners.

Location

This is the location to which the TRM will route output destined for this partner, unless routing is preempted by some other definition.

Interchange IDs and Qualifier

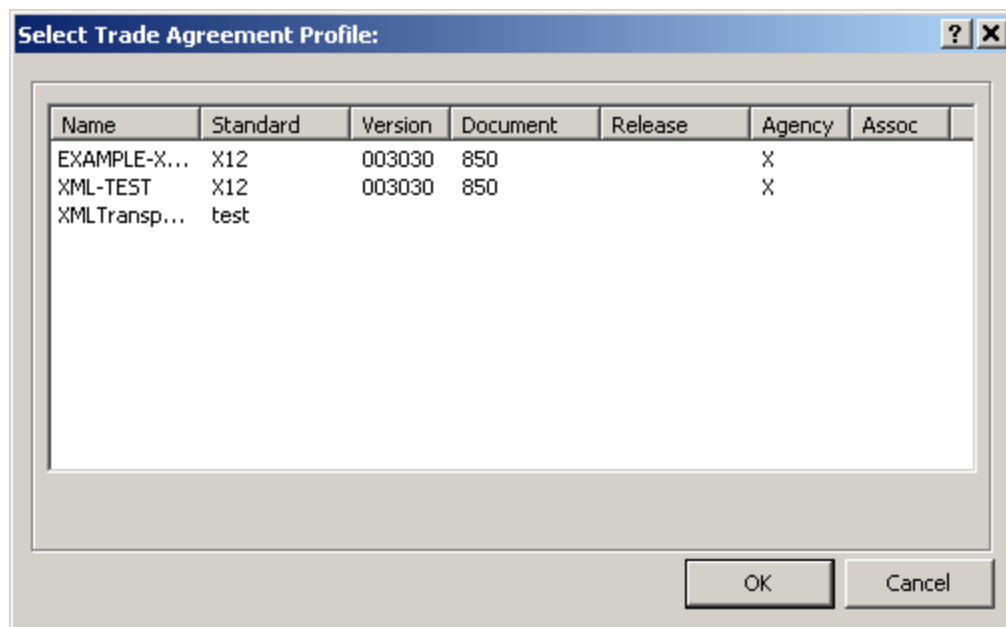
This group contains the interchange level IDs and qualifier for this partner. If you have selected a pre-defined partner, you will not be able to change this information until you change the partner name.

Functional Group IDs and Qualifier

This group contains the functional group level IDs and qualifier for partner. If you have selected a pre-defined partner, you will not be able to change this information until you change the partner name. Whether or not you see this level of information depends on whether or not the incoming envelope contains a second or functional group level wrapper.

(Partner Wizard) Select Trade Agreement Profile Page

When you select the **Add** button, a **Select Trade Agreement Profile** page appears. You must choose from among pre-defined profiles, and then select **OK** to add this to the trade agreement list for your partner.



Select Trade Agreement Profile (Partner Wizard)

(Partner Wizard) Trade Agreement Options Page

The trade agreement is essential for any document processing. These properties are associated with a partner relationship, a partner, or a standard ID definition.

For more information about how to complete this information, refer to the following topics:

- *Partner Trade Agreement Options Window* (on page 807)
- *Partner Relationship Trade Agreement Options Window* (on page 842)
- *Standard ID Trade Agreement Options Window* (on page 881)

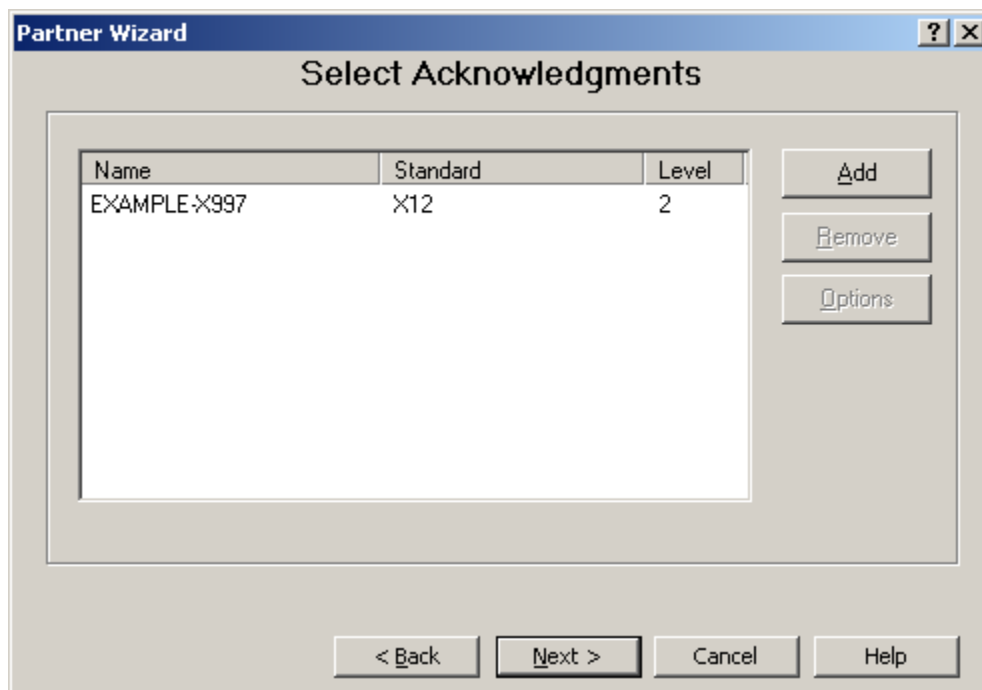
The screenshot shows a dialog box titled "Partner Trade Agreement Options: EXAMPLE-X850". At the top, there is an "Output:" field with a dropdown menu showing "1,EXAMPLE,1,FPO" and "1 of 1" to its right. Below this are three tabs: "Sending Partner" (selected), "Recipient Partner", and "Misc". The "Sending Partner" tab contains several input fields: "Partner Name:" with a dropdown, "Location:" with a text box, and "Override Location:" with a text box. There are two sections, "Interchange" and "Functional Group", each containing "ID:" and "Qual:" fields, and "Int ID:" and "Int ID2:" fields. At the bottom of the dialog are "OK", "Cancel", and "Apply" buttons.

Trade Agreement Options (Partner Wizard)

(Partner Wizard) Select Acknowledgments Page

This page allows you select an acknowledgment for this partner to be used for a sender that requires return acknowledgments. This list is associated with an acknowledgment selected for a partner relationship, a partner, or a standard ID definition, depending on which you are currently defining.

The TRM searches the list for acknowledgments whose wrapper matches that of the incoming document. For more information about how the TRM searches for matching acknowledgments, refer to the chapter, *Understanding Processing Flow* (on page 45).



Select Acknowledgments Page (Partner Wizard)

Acknowledgments List

You select a pre-defined acknowledgment from the **Select Acknowledgment Profiles** dialog box. The list displays information that should allow you to distinguish between profiles, such as the profile name, the standard to which it belongs, and the level of the acknowledgment. You can have multiple acknowledgment profiles. The TRM will generate any acknowledgments on the list that match the wrapper for the identified inbound standard that is the subject of the acknowledgment.

Add Button

To add acknowledgments to this partner profile, you select the **Add** button, and the **Select Acknowledgment Profile** dialog box appears.

Remove Button

The **Remove** button allows you to remove an acknowledgment profile from the list. This does not delete the profile; it only makes it unavailable as a possible choice for this partner.

Options Button

The **Options** button displays the **Acknowledgment Options** window, which allows you to specify alternative location addresses, and service characters, if the output is a delimited standard.

(Partner Wizard) Acknowledgment Options Page

The acknowledgment is essential for any control document or backward acknowledgment processing. These properties are associated with an acknowledgment selected for a partner relationship, a partner, or a standard ID definition, depending on which you are currently defining as determined by which trade agreement type you have selected.

For more information about how to complete this information, refer to the following topics:

- *(Partner) Acknowledgments Page* (on page 799)
- *(Partner Relationship) Acknowledgments Page* (on page 839)
- *(Standard ID) Acknowledgments Page* (on page 878)

The screenshot shows a dialog box titled "Partner Relationship Acknowledgment Options: EXAMPLE-X997". The "General" tab is active. Under "Send Outputs:", there are two empty text boxes labeled "From:" and "To:". Under "Service Characters:", the checkbox "Copy from input stream" is checked. Below it are seven empty text boxes for "Segment Terminator", "Component Delimiter", "Release Character", "Tag Delimiter", "Element Delimiter", "Decimal Mark", and "Repeat Separator". At the bottom are "OK", "Cancel", and "Apply" buttons.

General Page, Partner Acknowledgment Options (Partner Wizard)

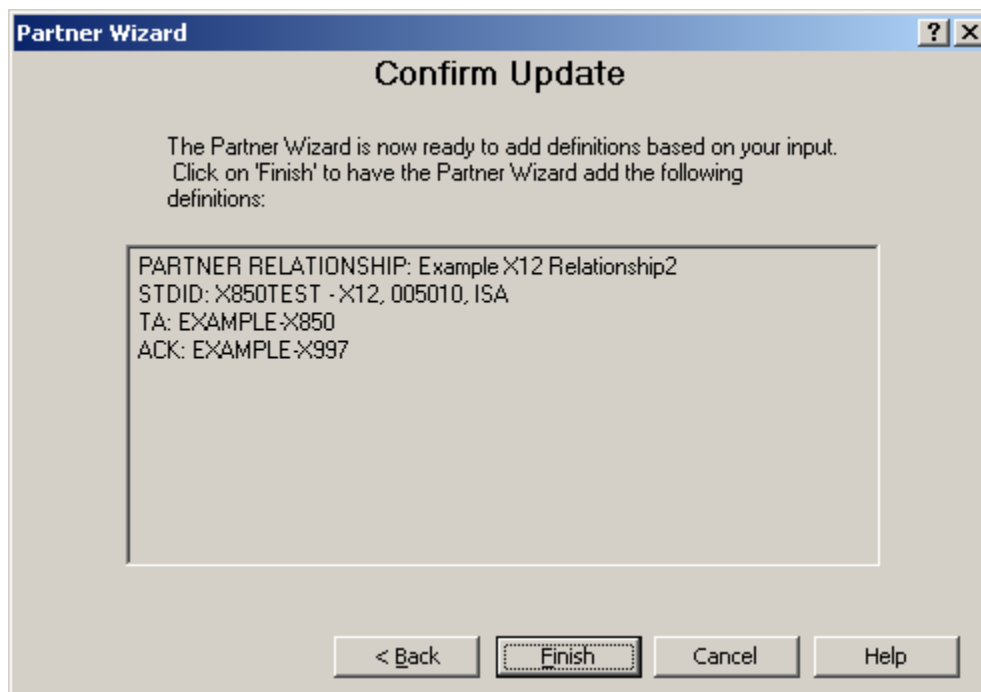
(Partner Wizard) Confirm Update Page

The **Confirm Update** page specifies the new partner relationships, partner(s), and partners locations that you will create using existing wrapper definitions, and the new links of partner relationships, partners, or standard IDs with existing trade agreements and acknowledgments.

This list indicates the new definitions that the wizard will add to Partner Explorer and the new links it will make with existing definitions. You must complete the process to add the definitions. If you have errors in your definitions, you will have an opportunity to correct the problems. If after you complete the wizard process you find that you need to make changes, you can do so easily from Partner Explorer.

Partner Relationship

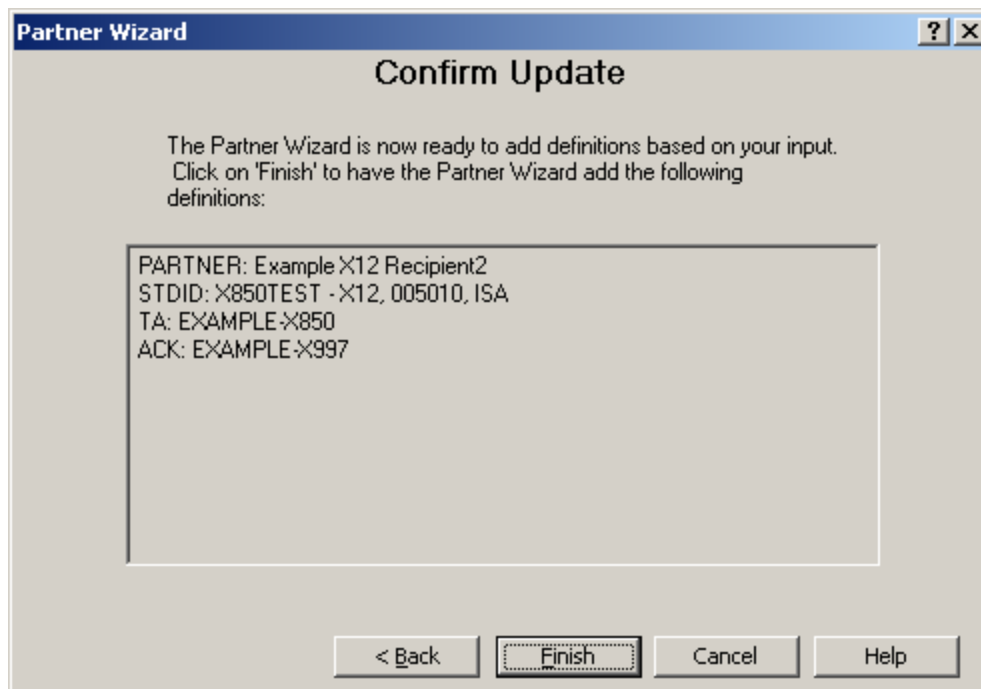
The following table shows you the definitions you can expect to see listed on this page if you have chosen the option **Control Processing Based on NEW Partner Relationship**. Some definitions may not appear, depending on what you have configured. Such definitions are listed as optional.



Definition	Notes
PARTNER RELATIONSHIP	New definition required. The combination of partners must be unique. If you create new partners for the relationship, those new partners will appear on the update list.
PARTNER	Optional. This creates a new partner if you have entered unique partner information for the sending or recipient partner.
LOCATION	This creates a new location if you have entered a unique location name. If you create a new location, you must also select a wrapper for standard ID.
STDID	If you create a new location, you will have to select a wrapper for Standard ID. This definition appears if you select a wrapper, whether the wrapper already exists for this location or not.
TA	This existing trade agreement was chosen for this partnership.
ACK	This existing acknowledgment was chosen for this partnership.

Partner

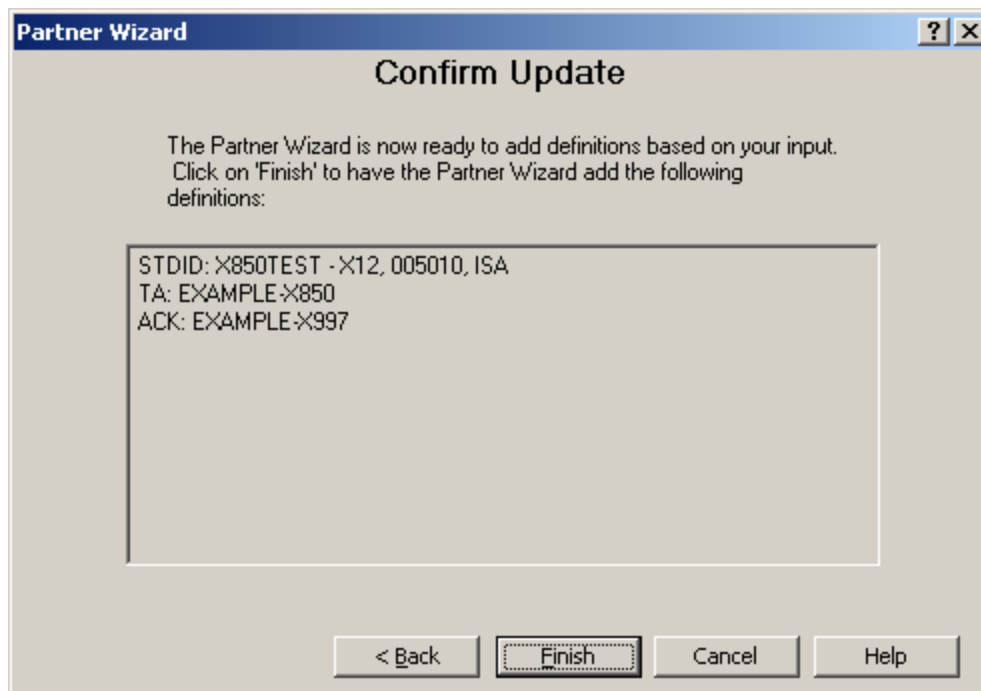
The following table shows you the definitions you can expect to see listed on this page if you have chosen the option **Control Processing Based on NEW Individual Partners**. Some definitions may not appear, depending on what you have configured. Such definitions are listed as optional.



Definition	Notes
PARTNER	New definition required.
LOCATION	This creates a new location if you have entered a unique location name. If you create a new location, you must also select a wrapper for standard ID.
STDID	If you create a new location, you will have to select a wrapper for Standard ID. This definition appears if you select a wrapper, whether the wrapper already exists for this location or not.
TA	This existing trade agreement was chosen for this partner to be used when the partner is a receiver.
ACK	This existing acknowledgment was chosen for this partner to be used when the partner is a sender.

Standard ID

The following table shows you the definitions you can expect to see listed on this page if you have chosen the option **Control Processing Based on NEW Incoming Wrapper Only (Standard ID)**. Some definitions may not appear, depending on what you have configured. Such definitions are listed as optional.



Definition	Notes
LOCATION	This creates a new location if you have entered a unique location name.
STDID	You must select a wrapper for Standard ID.
TA	This existing trade agreement was chosen for this standard ID wrapper to be used as a default without matching a partner.
ACK	This existing acknowledgment was chosen for this standard ID wrapper to be used as a default without matching a partner.

Finish Button

If these definitions are correct, select the **Finish** button.

Back Button

If you want to double-check for possible errors, select the **Back** button to check the information you have entered.

Cancel Button

If you want to cancel the process with no changes, select the **Cancel** button.

This page intentionally blank

Appendix Licenses

This section provides additional legal notices and information.

Xerces

This information includes both the Apache license and an Apache Xerces notice.

Apache License

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and

2. You must cause any modified files to carry prominent notices stating that You changed the files; and

3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

Apache Xerces Notice

NOTICE file corresponding to section 4(d) of the Apache License, Version 2.0, in this case for the Apache Xerces distribution.

This product includes software developed by
The Apache Software Foundation (<http://www.apache.org/>).

Portions of this software were originally based on
the following:

- software copyright (c) 1999, IBM Corporation.,
<http://www.ibm.com>.

This page intentionally blank

Glossary of Terms

A

ANSI

American National Standards Institute (ANSI) was founded in 1918 as the national coordinator for standards in the United States. ANSI charters committees, called Accredited Standards Committees (ASC), that focus on specific areas of standardization. It approves standards when development bodies present definitions that have been reached by consensus by its members. It also functions as a clearinghouse for national and international standards.

ASCII codes

ASCII is an acronym for American Standard Code for Information Interchange. A seven-bit code with an eighth bit for parity. This character code set is similar to the ANSI code set, but the ASCII set is used most often for character manipulation.

B

Bounded Loops

Bounded loops are loops, which are a group of repeating segments, that are delineated by a header segment to mark the beginning of the group and a trailer segment to mark the end of the group. Within X12 these segments are LS for the header and LE for the trailer.

C

Closed Trade Group

Closed trade groups allow only sending partners who are defined as members of the group, to send documents to an entity, either another partner or a default standard ID, whose trade agreement is also associated with the group. You add sending partners to groups by adding the group to the Group page of the partner definition. The partner is then listed in the Members container. You associate trade agreements with a group by specifying a group for a trade agreement on the list of a partner or Standard ID. Trade agreements associated with a group are used only when a valid trade agreement is not found on either the recipient partner or standard ID and the sending partner belongs to a group. This feature supports sender-based routing.

Conditional Condition

If the first data element specified is present, then all others must be present.

D

Data Element Store (DES)

The Data Element Store (DES) comprises the parsed input wrapper (WRAP) and its associated status, statistics and errors (SSEWRAP) and the parsed input document (DOC) and its associated status, statistics, and errors (SSEDOC). Users can control access to this information using Edibasic.

E

EDIFACT

EDIFACT is an acronym that stands for Electronic Data Interchange for Administration, Commerce and Transport. They are International Standards Organization (ISO) rules that comprise a set of standards, directories and guidelines for the electronic interchange of structured data between computerized data systems. These rules are under the aegis of the United Nations.

Exclusion Condition

Not more than one of the data elements specified may be present.

G

Global Variable

A global variable is a variable declared with a **Global** statement on a **Variables** page within user validation or mapping routines. A global variable is available only within the input stream in which it is declared. If a global variable is declared during the processing of one of many interchanges within an input stream, the variable is accessible from any subsequent interchange within that input stream, but not from another input stream.

I

Input stream

An input stream is all the data contained in a single logical file. The physical structure of that logical file can vary, as determined by the software controlling the data, such as an operating system (Windows) or a messaging system (MessageWay). An input stream might be a message, an attachment to a message, or a physical data file. If this input stream comprises EDI data, it may contain one or many interchanges. During processing all memory is initialized between input streams, so processing for each input stream is independent of all other input streams. This is important with regard to global variables, which are available only within the input stream in which they are declared.

L

Lifetime

Objects that represent data comprise variables and elements. These have both a lifetime and a scope. The lifetime of data elements for a document is until the data element stores of Doc and SSEDoc are cleared. The lifetime of data elements for wrappers is until the data element stores for WRAP and SSEWrap are cleared, for the next interchange. Memory is not automatically cleared between the translation of documents if the consecutive incoming sets are the same.

The lifetime of a global variable declared on a **Variable** page is from the point of declaration until the termination of processing of the input stream. The lifetime of a local variable declared on a **Variable** page is the duration of processing of the wrapper or document. The lifetime of a temporary variable defined (declared) within a method or a temporary variable created by Edibasic (undeclared) is the single execution of that method.

List Conditional Condition

If the first data element specified is present, then at least one of the others must be present.

Local Variable

A local variable is one that is declared with a **Dim** statement on a **Variable** page within user validation or mapping routines.

O

Open Trade

Open trade is a concept meaning that you may or may not know the sender; any trading partner can send data to a given recipient. This is in contrast to the more common concept of a partner relationship where both the sender and recipient must be defined and be part of a relationship. There are two types of open trade: when the recipient is defined, and when the recipient is not defined. If you want to match recipient IDs, you must create a partner profile for the recipient. If you do not need to match recipient IDs, you can specify a default trade agreement on the Trade Agreements page of the Standard ID window for the incoming standard listed within one of the Locations/StdID containers.

P

Paired Condition

If any data element specified is present, then all must be present.

R

Required Condition

At least one of the data elements specified must be present.

S

Scope

We use scope in two ways for different entities: elements and variables. For elements, scope is used to refer to the current nested repetition of segments and loops. For example, if you access an N4.2 element and do not specify the explicit occurrences using Edibasic, processing defaults to the current scope. This scope is set when you use visual mapping and by InitOcc, NextOcc, and ResetOcc functions within Edibasic.

The scope of a variable declared with a DIM or GLOBAL statement depends on the object for which it is declared. The scope of variables declared on a variable page for an object is all methods for that object and all subordinate objects. The scope of temporary variables or local variables declared within a method is that method. Although a variable exists for its lifetime, the scope set for the variable determines if Edibasic code can access it.

SWIFT

SWIFT is an acronym for the Society for Worldwide Interbank Financial Telecommunications. It is a consortium formed in 1973 by US and European banks. The primary purpose of the system and its related network services is the global transmission of a wide range of financial transaction messages in a secure, standardized format between users worldwide.

T

Temporary variable

A temporary variable is one declared on a **Method** page using a **Dim** statement or is one created by Edibasic when it cannot identify that variable name as a declared variable that is valid for the scope.

Translator Runtime Module (TRM)

The Translator Runtime Module is that part of the target (runtime or production mode) environment that does EDI processing. In contrast to the Workbench, TRMs only process data. You create configurations in the Workbench, where you can also test the configurations using the TRM. This common core allows you to test using the Workbench and process the data in a production environment.

X

X12

The ASC X12 is the accredited subcommittee of ANSI charged with the development of standard data formats for business transactions with cross-industry application. It was chartered in 1979 by ANSI to develop uniform standards for electronic interchange of business transactions.

Index

- • 707

&

& • 708

*

* • 707

/

/ • 707

+

+ • 706

<

< • 709

<= • 709

<> • 708

=

= • 708

>

> • 709

>= • 709

A

Abs function • 728

Acknowledgment

backward, alternative mailboxes • 276, 287

backward, generating • 276, 533

cross-reference file, linking • 579

cross-reference file, specifying • 130

document type • 578

expecting inbound, flag • 273, 284

inbound • 220

inbound, configuring trade agreement without mapping • 239

inbound, trade agreement options required for reconciliation • 223, 239

level • 579

setting flag in wrapper • 273, 284

specifying for a partner • 779

specifying for a partner relationship • 824

specifying on acknowledgment profile • 231

Acknowledgment profiles

conditions to generate acknowledgment • 535

configuration • 20

copying • 492

determining where to attach • 22

document reporting • 535

finding, description of TRM task • 57

inbound wrapper, specifying • 230

maps for document and wrapper • 534

matching, optional ERM task • 45

options, specifying • 232

overview • 219

report, printing • 234

security user exit • 537

security user exit, specifying • 233

summary document • 537

summary document, specifying • 233

text files, generating • 233

Acknowledgment, See also Control documents • 532

Acknowledgments

application, document type • 578

application, specifying document type • 123

backward, determining mailboxes for • 27

backward, generating • 99

backward, routing • 103

cross-reference file • 577

document type • 123

internal statuses • 236, 237

levels of response • 236, 577

methods to report information • 235

purpose • 236

receiving secured • 334

sending secured • 332

Standard ID default • 311

understanding • 235

Action

ERM task, specifying on trade agreement profile • 223

error types • 423

error, specifying on trade agreement profile • 223

Addition operator • 706

Alias

partner, defining • 284

specifying for trade agreement • 227

Aliases

outbound locations • 263

outbound partners • 74, 249, 257, 263

partner, outbound mailbox from • 91

And operator • 710

Area, for segment or loop in document • 563

Asc function • 729

Assignment error • 483

Audit user exit

example to log events • 330

Avg function • 729

B

Binary coded decimal (bcd)

element • 644

literal • 698

variable • 695

Binary data, element • 596

Breaking conditions

outbound document in separate interchange •
273, 284

outbound interchanges • 97

C

C++ Builder user exits • 330

Category

composite element • 553

document • 123, 577

document, specifying • 575

element • 127, 141, 596

segment • 125, 139, 635

standard version • 120, 649

standard version, specifying • 651

wrapper • 130, 679

wrapper, specifying • 675

Characters

between segments, ignoring during parsing •
652

delimiters • 683

list of, to ignore between segments during
parsing • 120, 651

number of, in string • 744

string, accessing leftmost • 743

string, accessing middle • 745

string, accessing rightmost • 750

valid, for data type • 119

Check Errors menu commands • 491

Chr\$ function • 731

Closed trade group

adding a partner to • 781

adding a trade agreement associated with
recipient partner to • 779

adding a trade agreement associated with
Standard ID to • 781, 859

creating • 780

deleting • 784

deleting a trade agreement associated with
recipient partner from • 783

deleting a trade agreement associated with
Standard ID from • 783

implications for processing control • 21

trade requirement criteria • 23

Codes

adding element • 593

element ID • 589

element, automatic validation • 594

element, defining • 126, 140

element, deleting • 594

element, partitioned • 590

element, validate for document • 123, 576

element, validate for standard version • 120, 651

element, validate for wrappers • 130, 675

Commands

new functionality • 1

shortcuts for Test menu • 762

shortcuts for Workbench menus • 487

Show Fields • 616, 625

Show Repeats • 616

Test menu • 762

Workbench menus • 487

Compliance checking, inbound data • 324

Component elements

adding • 544

deleting • 545

delimiter character • 685

modifying • 545

requirement in composite • 541

sequence number as part of element name • 698

sequential number in composite element • 731

validate elements in a composite • 326

validate routines, adding • 546

validate routines, declaring variables • 547

Component\$ function • 731

Composite elements

adding component elements • 544

adding to a standard version • 542

conditions • 551

conditions, adding • 549

conditions, deleting • 550

configuration window • 538

deleting • 543

deleting component elements • 545

displaying all sequentially • 544

ID • 623

list of segments where used • 554

modifying • 543

occurrence as part of element name • 698

repeating in segments • 629

repetition separator • 685, 816, 822, 851, 856, 890, 895

saving • 543

validate routines, adding • 548

validate routines, declaring variables • 547

validate routines, printing • 586

Concatenation operator • 708

Condition method • 712

Conditional statement

case statement • 726

- conjunction operator • 710
 - disjunction operator • 710
 - do statement • 719
 - equality operator • 708
 - equivalence operator • 711
 - exclusion operator • 710
 - greater than operator • 709
 - greater than or equal operator • 709
 - If statement • 723
 - implication operator • 711
 - inequality operator • 708
 - less than operator • 709
 - less than or equal operator • 709
 - Not operator • 706
 - While statement • 728
- Conditions
- composite elements • 551
 - composite elements, adding • 549
 - composite elements, deleting • 550
 - Edibasic method for • 712
 - Edibasic statement for • 719, 723, 726, 728
 - Edibasic, to create output • 212
 - mapping example • 175
 - mapping, typical use • 584
 - segment • 637
 - segment, adding or modifying • 633
 - segment, deleting • 634
 - types • 552, 638
- Configurations
- acknowledgment profiles • 532
 - changing location of CFG directory • 519
 - copying within and between Workbench environments • 412, 492
 - determining from requirements • 12
 - exporting to TRN file • 412, 497, 771
 - files for testing • 389
 - generating text files • 499, 501
 - importing from TRN file • 501
 - importing to TRN file • 416
 - links to internal fields • 50
 - moving • 411
 - null wrapper example • 146
 - planning • 11
 - purpose • 12
 - task tree, basic • 16
 - translation example, list of • 40
 - translation example, review of • 38
 - use • 17
 - user exit AUTACK example, list of • 335

- Conjunction operator • 710
- Const statement • 718
- Contents • See also, Documents
 - defined • 120
 - tasks to define • 121
 - translate option • 36
 - TRM task to scan header • 61
 - validate, option • 31, 33
 - validation of, and routing • 86
 - validation, TRM task • 68
- Control document, See Acknowledgment • 532
- Control documents
 - configurations to receive CONTRL secured with AUTACK • 352
 - configurations to send CONTRL secured with AUTACK • 337
 - EDIFACT CONTRL secured with AUTACK • 332
- Control number • 55
- Control reference
 - filename used on processing report • 70
 - inbound document, on processing report • 70, 74
 - inbound wrapper, on processing report • 57, 74
 - inbound, identified on processing report • 55
 - user exit, configuration requirement • 330

- Conventions, typographical • 1
- Conversion
 - ANSI numeric code to ASCII character • 731
 - database, during install • 10
 - format non-string data to string date or number • 737
 - format string expression to date or number • 732
- Convert\$ function • 732
- Copy command
 - copy configuration definitions to databases • 495
 - options • 492
- Count function • 732
- Cross-references
 - acknowledgment • 130, 577, 679
 - adding tables • 557
 - copying • 492
 - deleting tables • 558
 - error code table, link to acknowledgment • 579
 - look-up for string • 759
 - modifying tables • 557
 - saving • 558
 - tables • 556
- D**
- Data
 - characters to ignore during parsing • 120

existence of element, determining • 742

null string, in internal field • 53, 55, 61, 252

Data Element Store

location as part of element name • 698

option to view contents on processing report • 388, 508

SetField function to override stored value • 325

viewing contents on processing report • 512

Data types

base types for formatting • 119, 643

binary coded decimal (bcd) element • 644

binary coded decimal (bcd) literal • 698

binary coded decimal (bcd) variable • 695

EBCDIC • 644

Edibasic literals • 697

Edibasic variables • 695

element • 589, 698

errors during generate • 483

integer literal • 698

integer variable • 695

standard version • 641

standard version, defining • 646

standard version, defining BCD • 647

standard version, defining EBCDIC • 646

standard version, modifying • 647

string literal • 698

string variable • 695

valid characters • 119, 644

variables, determining • 758

variant variable • 695

Databases

adding to Select Environment list • 527

changing location of DB directory • 519

configuration, maintaining • 419

copying specific definitions • 495

current environment • 487

exporting specific definitions • 498, 771

importing specific definitions • 505

loading standard definitions • 507

maintaining configuration • 522

removing from Select Environment list • 527

switching between, for testing • 529

DateSerial function • 733

Day function • 733

Decimal mark character • 686

Delimited data

alternative service characters for backward acknowledgment • 276, 287

alternative service characters for output • 273, 284

composite element • 553

- document • 123, 578
- element • 127, 141, 596
- segment • 125, 139, 636
- service characters • 130
- standard version • 120, 650
- wrapper • 130, 680

Delimiters, See Service characters • 130

DES, See Data Element Store • 325

Digital signature

- configurations for user exit generation • 357
- configurations for user exit validation • 344
- user exit example • 337
- user exit to generate • 347
- user exit to validate • 363

Dim statement • 718

Directory

- changing location of root • 519
- environment • 514

Disjunction operator • 710

Division operators • 707, 708

Do statement • 719

Document

- adding • 568
- configuration window • 559
- deleting • 569
- Edibasic validate routines • 122
- EDIFACT AUTACK • 331
- EDIFACT CONTRL, secured with AUTACK • 332
- inbound, on processing report • 61
- inbound, specifying on trade agreement profile • 225
- loop, adding • 570
- loop, deleting • 571
- loop, modifying • 571
- map, creating • 171
- map, generating text files • 182
- modifying • 569
- outbound map, selected for trade agreement • 660
- outbound, filename on processing report • 70
- outbound, in separate interchange • 273, 284
- outbound, on processing report • 70
- outbound, specifying on trade agreement • 227
- properties, defining • 123
- report options • 559
- report, printing or previewing • 143, 571
- saving • 569
- security, configurations for AUTACK example • 339
- security, example with user exit to create hash value • 358
- security, specifying for input stream • 225

- segment, adding • 570
 - segment, deleting • 571
 - segment, modifying • 571
 - simple type • 123, 578
 - summary, configurations for AUTACK example • 354
 - summary, example with user exit to create hash value • 346
 - summary, map for outbound • 662
 - summary, specifying for acknowledgment profile • 233, 537
 - summary, specifying for trade agreement • 227
 - text file, generating • 127, 572
 - type, specifying • 576
 - validate element codes • 577
 - validate routines, adding • 573
 - validate routines, adding to segments • 572
 - validate routines, declaring variables • 573
 - validate routines, declaring variables on segments • 574
 - XML, defining • 373
- Document reject
- error action selected on trade agreement • 656
 - user-invoked, during parsing • 325
- Document-level break • 273, 284
- Documents • 31
- copying • 492
 - creating outbound, TRM task • 70
 - defining • 122
 - displaying all sequentially • 570
 - outbound, grouping in same interchange • 97, 227
 - rejecting • 86
 - status of inbound, on processing report • 70, 74
 - TRM task to scan header • 61
 - types of, defining • 123
- DTD, See XML data • 374
- E**
- EDI Runtime Module
- character sets, defining • 646
 - support • 644
- Edibasic
- code flag on component element in segment • 540
 - code flag on element in segment • 624
 - code flag on segment or loop in document • 564
 - code flag on segment or loop in wrapper • 666
 - code to execute first • 583

- code, using Print statement to debug • 725
- conditions to create output • 212
- data types for literals • 697
- data types for variables • 695
- edit window • 580
- introduction • 693
- keywords • 693
- loop and segment occurrences, controlling • 584
- mapping instructions, adding at document or wrapper level • 613
- mapping instructions, adding at element level • 615
- mapping instructions, adding at loop, segment, or composite element level • 614
- mapping outbound location • 91
- mapping outbound partner IDs • 74
- mapping techniques • 157
- operators, descriptions • 704
- operators, list • 704
- operators, syntax • 706
- order executed in wrapper and document maps • 70
- reasons for using • 165
- remarks in code • 726
- syntax errors, checking • 585
- syntax requirements • 693
- user exit to generate digital signature • 347
- user exit to validate digital signature • 363
- user exits • 327
- validate routines on Document window • 122
- validate routines on Element window • 126, 140
- validate routines on Segment window • 124, 136
- validate routines on Wrapper window • 129
- validate routines, overview • 323

Edibasic functions • 728

- Abs, reference • 728
- Asc, reference • 729
- Avg, reference • 729
- Chr\$, reference • 731
- Component\$, reference • 731
- Convert\$, reference • 732
- Count, reference • 732
- DateSerial, reference • 733
- Day, reference • 733
- Element\$, reference • 734
- exception to invoke document reject during parsing • 325
- Exception, reference • 734
- Field\$, reference • 735
- Format\$, reference • 737
- Hour, reference • 740
- InitOcc, reference • 741

-
- InStr, reference • 741
 - IsNull, reference • 742
 - IsNumeric, reference • 742
 - LCase\$, reference • 743
 - Left\$, reference • 743
 - Len, reference • 744
 - LTrim\$, reference • 744
 - Max, reference • 744
 - Mid\$, reference • 745
 - Min, reference • 745
 - Minute, reference • 746
 - Month, reference • 746
 - NextOcc, reference • 747
 - Now, reference • 748
 - occurrence, reference • 748
 - ResetOcc, reference • 749
 - Right\$, reference • 750
 - RTrim\$, reference • 751
 - Second, reference • 751
 - SetField to map mailboxes • 214
 - SetField, reference • 751
 - Space\$, reference • 753
 - Str\$ to convert integer to string • 178
 - Str\$, reference • 754
 - StrComp, reference • 754
 - String\$, reference • 754
 - Sum, reference • 755
 - syntax • 728
 - TimeSerial, reference • 755
 - Trim\$, reference • 756
 - UCase\$, reference • 756
 - User, reference • 756
 - Val, reference • 757
 - Value, reference • 757
 - VarType, reference • 758
 - Weekday, reference • 759
 - Xref\$, reference • 759
 - Xref\$, XrefR\$ to invoke cross-reference • 556
 - XrefR\$, reference • 759
 - Year, reference • 760
- Edibasic methods
- Condition, at document level • 613
 - Condition, at element level • 615
 - Condition, at loop, segment, or composite element level • 614
 - Condition, reference • 712
 - GetNext page • 584
 - GetNext, at loop, segment or composite element level • 614
 - GetNext, reference • 713

- limit on code per method • 157, 580
- MapEle example using Str\$ • 178
- MapEle page • 585
- MapEle, at element level • 615
- MapEle, reference • 714
- mapping condition, typical use • 584
- Start, at document level • 613
- Start, at loop, segment, or composite element level • 614
- Start, reference • 715
- Stop page • 584
- stop processing • 729
- Stop, at loop, segment or composite element level • 614
- Stop, reference • 715
- syntax • 712
- Validate page • 585
- Validate, reference • 716
- ValidateAll page • 585
- ValidateAll, reference • 717

Edibasic statements

- Const, reference • 718
- Dim, reference • 718
- Do-Loop, reference • 719
- Exit, reference • 729
- For Each, reference • 720
- For, reference • 730
- Global, reference • 721
- If, reference • 723
- Let, reference • 724
- Mid\$, reference • 724
- Print, reference • 725
- Rem, reference • 726
- Select Case, reference • 726
- syntax • 717
- While, reference • 728

Edibasic user exit example • 330

Edibasic, See also Mapping instructions, Validate routines, Variables • 693

EDIFACT

- AUTACK security document example • 331
- AUTACK, summary document • 227, 233
- CONTRL document secured with AUTACK • 332
- CONTRL inbound secured with AUTACK • 353
- CONTRL outbound secured with AUTACK • 338
- data element tag • 539, 588, 623
- PAYMUL inbound secured with AUTACK • 338
- PAYMUL outbound secured with AUTACK • 353
- PAYMUL secured with AUTACK, list of definitions • 350, 364
- standards definitions supported • 7

translation tutorial • 9

Edit menu commands • 489

Element

as a tag in fixed-length segment • 136

assigning to internal field • 630

average of values • 729

configuration window • 587

delimiter character • 684

Edibasic method to generate • 714

Edibasic validate routines • 126, 140

fixed-length data, not padded, variable • 596

fixed-length data, padded • 596

ID • 623

list of codes • 126, 140

list of segments where used • 127, 141

names, syntax • 698

occurrence as part of element name • 698

occurrences, counting in loop or segment •
732

properties, defining • 127, 141

sequence number as part of element name •
698

simple, validate • 325

value, access during parsing • 734

Element\$ function • 734

Elements

adding to a standard version • 590

codes, adding • 593

codes, automatic validation • 594

codes, deleting • 594

component • 538

composite • 538

deleting • 591

displaying all sequentially • 592

document, defining • 126

Edibasic method to generate • 585

list of segments where used • 599

missing or truncated trailing, in fixed-length
standard • 153

modifying • 591

repeating in segments • 629

repetition separator • 685, 816, 822, 851, 856,
890, 895

saving • 591

validate codes for document • 123, 576, 577

validate codes for standard version • 120,
649, 651

validate codes for wrappers • 130, 675, 679

validate routines, adding • 592

validate routines, declaring variables • 593

validate routines, printing • 586

wrapper, assigning to internal fields • 136

wrapper, defining • 140

Envelope • 31

defined • 127

naked interchange or null wrappers • 128

tasks to define • 128

Environment

adding database to Select Environment list • 527

configuration, checking for Example • 38

configurations, moving • 411

creating and using • 406

current database • 487

deleting • 411

file locations • 514

installing a new • 527

list of user exits • 520

moving • 410

removing database from Select Environment list • 527

subdirectories and files • 525

switching • 529

Equality operator • 708

Equivalence operator • 711

Eqv • 711

Error codes

ranges for TRM types • 423

TRM, list of • 425

Errors

action for, list of • 656

action for, specifying on trade agreement profile • 223

actions for various types • 423

Edibasic code, using Print statement to debug • 725

Edibasic syntax, checking • 585

generating text files • 483

orphaned records • 604

predefined compliance • 423

processing report • 469

system • 423

troubleshooting • 423

user-defined • 423

Errors, See also Exceptions • 423

Examples

element name, syntax • 698

mapping, condition • 176, 216

mapping, drag-and-drop • 175

mapping, for beginners • 159

- mapping, literal • 164, 176
- mapping, local variable • 178
- mapping, source and destination mailboxes • 213, 214
- Standard ID configurations • 316
- translation • 9
- translation configurations, finding • 38
- translation, list of configurations • 40
- translation, review of configurations • 38
- translation, running a test • 41
- translation, viewing test reports • 43
- translation, XML • 368
- tutorial to create X12 or EDIFACT • 9
- user exits • 9
- Exception function • 734
- Exceptions
 - actions, invoke for • 734
 - limiting number on processing report • 513
 - reject for required partner IDs • 53
 - runtime • 423
- Exceptions, See also Errors • 423
- Exclusion operator • 710
- Exit statement • 729
- Export command
 - options • 497
 - transfer configuration definitions to a TRN file • 771
 - use during testing • 773
- F**
- Features • 3
 - current • 7
- Field\$ function • 735
- File menu commands • 488
- File names
 - option to report • 508
 - reporting on processing report • 512
- Files
 - changing location of FILES directory • 519
 - configuration, for testing • 389
 - names generated for documents • 123
 - output, grouping documents in same interchange • 97, 227
 - output, rules for breaking • 97
 - test output, automatically deleting • 389
 - TRN, database version compatibility • 416
 - TRN, exporting configurations • 412
 - TRN, exporting from test • 403
 - TRN, importing configurations • 416
- Files, See also Input, Output, Text files • 609
- Fixed-length data
 - composite element • 553

- document • 123, 578
- element • 127, 141, 596
- element, in a delimited standard • 596
- parsing requirements • 474
- segment • 125, 139, 636
- segment tag, specifying for non-initial element • 616, 624, 630
- standard version • 120, 650
- variable-length segments • 153
- wrapper • 130, 680

For Each statement • 720

For statement • 730

Format\$ function • 737

Function syntax • 728

Functions, See also Edibasic functions • See also, Edibasic functions

Functions, See also Edibasic Functions • 329

- C language, for user exits • 329

- user exit, example to calculate hash value • 337

- user exit, example to generate digital signature • 337

- user exit, example to validate digital signature • 337

G

Generate command

- generating a single configuration • 501

- generating all definitions • 500

- options • 499

Generate menu commands • 490

GetNext method • 713

Global statement • 721

Global variable

- declaring • 696

- description • 696

- example for user exit • 344, 347, 357, 363

- lifetime • 697

- scope • 697

- statement to declare • 721

- to access input trailer data during mapping • 213

Greater than operator • 709

Greater than or equal operator • 709

Group

- defined in a document • 559

- mapping to different segments • 176

Group, See also Loop • 175

H

Hash value, user exit calculation • 346, 358

Help menu commands • 491

Hour function • 740

I

Icons

- Test menu commands • 762

- Workbench commands • 487

- ID codes, See Codes • 126
- Identification
 - partner • 251
 - standard, description of task • 47
 - standard, mandatory TRM task • 45
 - standard, using source location • 47
 - standard, using wrapper content • 47, 858
- If statement • 723
- Imp • 711
- Implementation guide defined • 116
- Implication operator • 711
- Import command
 - options • 501
 - transfer definitions from a TRN file • 505
- Inequality operator • 708
- Initialization of variables • 583
- InitOcc function • 741
- Input
 - changing location of IN directory • 519
 - characters to ignore between segments • 649
 - cross-reference • 556
 - document and security options • 655
 - filename on processing report • 47
 - parsing schema for composite element • 553
 - parsing schema for document • 123, 577
 - parsing schema for element • 127, 141, 596
 - parsing schema for segment • 125, 139, 635
 - parsing schema for standard version • 120, 649
 - parsing schema for wrappers • 130, 679
 - routing after validating contents • 33
 - routing after validating wrapper • 31
 - segments, marking end • 130
 - validation, Edibasic method • 716, 717
- Input stream
 - security document • 655
 - security user exit • 225, 655
- Installation
 - database conversions • 10
 - general • 10
 - new database environment • 10
- InStr function • 741
- Integer
 - example, converting to string • 178
 - literal • 698
 - variable • 695
- Interchange
 - naked, defined • 128
 - outbound, breaking conditions • 97
 - output documents in separate • 273, 284

rejecting • 86

routing • 86

unit of work for TRM • 46

Internal fields • 155

assigning to elements • 630

assigning to wrapper elements • 136

decision matrix to match with trade
agreement profile fields • 61

initialization based on inbound data • 53

links to configurations • 50

matching with trade agreement profile fields •
61

null strings • 53, 55, 61, 252

partner matching • 251

Show Fields command on Segment window •
616, 625

values, accessing using Edibasic • 735

values, storing using Edibasic • 751

IsNull function • 742

IsNumeric function • 742

K

Keywords for Edibasic • 693

L

LCase\$ function • 743

Left\$ function • 743

Len function • 744

Length

maximum element • 589

maximum element, for BCD data • 589

minimum element • 588

Less than operator • 709

Less than or equal operator • 709

Let statement • 724

Levels

acknowledgment • 99, 579

application acknowledgment • 577

Limits

Edibasic method, compiled code • 157, 580

number of user exits • 337, 520

number of variables • 696

Literal

data types • 697

example, mapping source • 175

Load Standards command

loading public • 507

options • 506

Local variable

declaring • 696

description • 696

lifetime • 697

scope • 697

statement to declare • 718

Locations

alternative, for backward acknowledgment •
276, 287

default • 17

for backward acknowledgments, determining
from configurations • 27

inbound • 17

inbound, as source station • 47

inbound, on processing report • 47

inbound, using to identify standard • 47

mapping source and destination • 214

outbound, alternate • 273, 284

outbound, for translation output • 91

outbound, for validation only • 91

outbound, mandatory TRM task to identify •
45

outbound, mapping • 25, 26

outbound, on partner definitions • 22

outbound, override by mapping • 91

outbound, override location • 246, 257

outbound, specifying • 25

outbound, using partner aliases • 263

output • 25

output, on processing report • 70

processing steps to find, for backward
acknowledgments • 27

processing steps to find, for translated output
• 26

processing steps to find, for validated output •
26

routing mailboxes, outbound • 25

source, creating • 303

source, on processing report • 47

Standard ID routing • 309

Loop

adding to a document • 570

area in document • 563

bounded, generating • 568

defined in a document • 559

deleting • 571

level in document • 563

mapping to different segments • 176

modifying • 571

- occurrence as part of element name • 698
- occurrences, controlling with Edibasic • 584
- occurrences, Edibasic Do-Loop statement • 719
- occurrences, Edibasic For Each statement • 720
- occurrences, Edibasic For statement • 730
- occurrences, Edibasic method to initialize • 715
- occurrences, Edibasic method to process • 713
- occurrences, Edibasic method to reset • 715
- occurrences, Edibasic While statement • 728
- requirement in document • 568
- versus segment, defining • 563

Loop, See also Group • 175

LTrim\$ function • 744

M

Maintenance

- configuration databases • 419
- packing database • 522

Map

- acknowledgment, specifying on acknowledgment profile • 231
- configuration window • 600
- creating a new • 607
- deleting • 608
- destination document, changing • 604

- document, creating • 171
- document, development guidelines • 111
- document, specifying on trade agreement • 227
- modifying • 608
- outbound document, on processing report • 70
- outbound document, selected for trade agreement • 660
- outbound summary document, selected for trade agreement • 662
- outbound wrapper, on processing report • 70
- outbound wrapper, selected for trade agreement • 660
- printing report • 609
- saving • 608
- source statement • 606
- text files, generating • 609
- types • 601
- using EDI standards • 107
- wrapper, creating • 168
- wrapper, development guidelines • 111
- wrapper, specifying on acknowledgment profile • 231
- wrapper, specifying on trade agreement • 227

MapEle method • 714

Mapping instructions

- accessing trailer wrapper data • 213

- conditions to create output • 212
 - conditions, reference • 712
 - conditions, statement for • 719, 723, 726, 728
 - deleting • 613
 - document or wrapper level • 613
 - Edibasic, adding at element level • 615
 - element, method to generate • 714
 - example, from loop/group to different segments • 176
 - example, literal button • 164
 - examples for beginners • 159
 - finding orphaned records • 608
 - literal to an element • 610, 611
 - loop, segment, or composite element level • 614
 - loops and segments, initialization • 715
 - loops and segments, processing occurrences • 713
 - loops and segments, resetting • 715
 - orphaned records, cause • 604
 - Print statement to debug • 725
 - remark statement to document code • 726
 - source and destination locations • 214
 - stop processing • 729
 - string variable, selecting part of • 724
 - technique, delete or erase • 164
 - technique, internal field • 164
 - technique, literal • 164
 - technique, once • 164
 - techniques, Edibasic • 157, 165
 - techniques, visual • 155, 161
 - variable, assigning value to • 724
 - visual, default behavior • 163
 - wrapper field to document element • 610
- ### Maps
- basic instructions for creating • 167
 - copying • 492
 - Edibasic user exit • 327
 - order of execution • 70
 - overview • 155
- ### Matching criteria
- default selection for inbound wrapper • 47
 - default, adding to wrapper for standard identification • 676
 - default, specifying • 130
 - deleting • 859
 - inbound wrapper • 18, 304, 687
 - not using • 18
- ### Matching strategy
- acknowledgment profile, specifying • 230
 - partners • 251

trade agreement profile • 222, 653

Max function • 744

Menus

Check Errors • 491

Edit • 489

File • 488

Generate • 490

Help • 491

Tools • 490

View • 489

Windows • 491

Message

receiving secured • 332, 337

sending secured • 334, 352

Message, See also Document • 122

MessageWay Solutions

web site • 6

Method Condition as Integer • 712

Method GetNext as Integer • 713

Method MapEle as String • 714

Method Start • 715

Method Stop • 715

Method Validate as Integer • 716

Method ValidateAll as Integer • 717

Methods, See also Edibasic methods • 175

Mid\$ function • 745

Mid\$ statement • 724

Min function • 745

Minute function • 746

Mod • 708

Modify Options, See Options command • 507

Month function • 746

Multiplication operator • 707

N

Negation

logical operator • 706

operator • 707

-
- NextOcc function • 747
 - Non-repudiation • 331
 - Not operator • 706
 - Notation conventions • See also, Conventions, typographical
 - Now function • 748
 - Null wrapper • 146
 - Numeric expression
 - absolute value • 728
 - convert to string value • 754
 - value of, convert from string value • 757
 - value of, convert from string value and apply format • 757
 - O**
 - Occurrence function • 748
 - Occurrences
 - access next • 747
 - initialize • 741
 - loops and segments, Edibasic method to initialize • 715
 - loops and segments, Edibasic method to process • 713
 - loops and segments, Edibasic method to reset • 715
 - maximum segment • 568
 - reset • 749
 - Operators
 - & • 708
 - * • 707
 - / • 707
 - + • 706
 - < • 709
 - <= • 709
 - <> • 708
 - = • 708
 - > • 709
 - >= • 709
 - And • 710
 - descriptions • 704
 - Eqv • 711
 - Imp • 711
 - list • 704
 - Mod • 708
 - Not • 706
 - Or • 710
 - syntax • 706
 - Xor • 710
 - Options
 - acknowledgment profile, specifying • 232
 - processing report • 508
 - trade agreement profile, specifying • 223

validation and translation • 31

Options command • 507

Or operator • 710

Output

adding line breaks during testing • 763

changing location of OUT directory • 519

cross-reference • 556

deleting automatically • 763

Edibasic conditions to create • 212, 584

generation schema for composite element • 553

generation schema for document • 123, 577

generation schema for element • 127, 141, 596

generation schema for segment • 125, 139, 635

generation schema for standard version • 120, 649

generation schema for wrappers • 130, 679

rejected documents • 86

routing, default on Standard ID • 309

routing, for translate option • 36

routing, for validate wrappers option • 31

rules for breaking • 97

segments, marking end • 130

Outputs

alternative IDs and locations • 273, 284

automatically deleting after testing • 389

grouping documents in same interchange • 97, 227, 660

list of, for specified input • 658

order of processing multiple • 97, 226, 658

specifying on trade agreement profile • 226

P

Pack Database command • 521

Partner definition

adding a new definition • 778

adding an alias to • 785

copying • 492

create with Partner Wizard • 294

creating • 267, 279

deleting a • 778

deleting an alias from • 785

implications for processing control • 21

list of acknowledgment profiles • 57

requirements on Standard ID • 308

specifying a trade agreement for • 779

specifying an acknowledgment for • 779

trade requirement criteria • 23

Partner Explorer

tasks vs. Partner Wizard tasks • 290, 313

using • 244, 297

Partner IDs

defining • 254

- inbound, matching for security • 22
 - inbound, no validation of • 31
 - inbound, on processing report • 55
 - inbound, validate recipient only • 33
 - inbound, validate sender and recipient • 36
 - inbound, validation task • 53
 - links to internal fields • 50
 - outbound, alternate • 246, 257, 273, 284
 - outbound, alternatives • 22
 - outbound, override by mapping • 74
 - outbound, TRM task • 74
 - outbound, values • 74
 - processing step to find definitions • 24
- Partner relationship
- identified on processing report • 57
 - TRM search task • 57
- Partner Relationship definition
- adding a • 823
 - completing a • 824
 - copying • 492
 - create with Partner Wizard • 292
 - creating • 267
 - deleting a • 824
 - implications for processing control • 21
 - list of acknowledgment profiles • 57
 - specifying a trade agreement for • 825
 - specifying an acknowledgment for • 824
 - trade requirement criteria • 23
- Partner Wizard
- partner relationship, create to control processing • 292
 - partner, create to control processing • 294
 - processing control decisions for partners • 291
 - processing control decisions for standard ID • 314
 - standard ID, create • 315, 316
 - tasks vs. Partner Explorer tasks • 290, 313
 - using • 250, 299
- Partners
- aliases for outbound • 74, 249, 257, 263, 663
 - aliases for, outbound location from • 91
 - criteria to define • 21
 - default • 266
 - definitions, configuring • 251
 - definitions, optional • 22
 - definitions, requiring • 266
 - inbound, partner relationship search task • 57
 - inbound, profile search task • 55
 - internal fields, using to match • 251
 - processing control • 257

reasons to define • 22, 243

Post-processing user exit • 330

Pre-processing user exit • 330

Print command • 522

Print Preview command • 524

Print statement • 725

Print to PDF Command • 525

Processes

list of TRM tasks • 46

mandatory TRM tasks • 29, 45

software release, creating • 112

test and production, integration • 15

Processing control

Partner definition, create with Partner Wizard
• 291, 294

partner definitions, requiring • 266

Partner Relationship definition, create with
Partner Wizard • 292

Standard ID definition, create with Partner
Wizard • 315

Standard ID definition, creating with Partner
Wizard • 314

Standard ID definitions and tasks • 307

types and implications • 21, 23

using default partner definitions • 266

using partner definitions • 257

using Standard ID instead of partners • 257

Processing flow

control • 17

overview • 30

Processing options

list of actions and error actions • 656

list of outputs • 658

validation and translation • 31

Processing report

control reference filename • 70

control reference of inbound document • 70,
74

control reference of inbound wrapper • 74

controlling information reported • 512

creating or suppressing • 512

date and time stamp, beginning • 47

definition where trade agreement was found •
61, 74

definitions used to create documents • 70

errors, troubleshooting • 469

inbound control reference • 55, 57

inbound document definition file • 61

inbound location • 47

inbound partner IDs • 55

inbound standard wrapper • 47

inbound wrapper file • 55

information options, setting • 388

limiting number of exceptions • 513

- locations for output • 70
- output filename • 70
- partner relationship for inbound partners • 57
- source mailbox • 47
- status of inbound document • 70, 74
- status of inbound wrapper • 74
- trace of parsing and generate processes • 472
- TRM version and build • 47
- understanding • 394
- values used to find trade agreement • 61, 74

Properties

- document, defining • 123
- element, defining • 127, 141
- segment, defining • 125, 139
- standard version, defining • 120
- trade agreement output, specifying • 227
- wrapper, defining • 130

Purpose

- configurations • 12
- MW Translator Workbench • 2

R

Reconciliation

- expecting inbound acknowledgment flag • 273, 284
- trade agreement options required for inbound acknowledgment • 223, 239

- Register function for user exits • 329

Reject

- document, user-invoked during parsing • 325
- exception for required partner IDs • 53
- security • 423

- Release character • 686

- Rem statement • 726

- Remarks in Edibasic code • 726

- Repetition separator • 685, 816, 822, 851, 856, 890, 895

Reports

- acknowledgment profile, printing • 234
- changing location of RPT directory • 519
- document, options • 559
- document, printing or previewing • 571
- Edibasic code used during parsing • 586
- example map • 183
- example test • 43
- map, printing • 609
- Standard ID, printing • 312
- trade agreement profile, printing • 229
- types • 522
- wrapper, options • 664
- wrapper, printing or previewing • 674

Requirements

- configurations • 12
- parsing inbound wrappers • 50
- partner definitions • 308
- trading • 11

ResetOcc function • 749

Right\$ function • 750

RTrim\$ function • 751

S

Second function • 751

Security document

- inbound • 655
- inbound, configurations for AUTACK example • 339
- specifying on trade agreement profile • 225
- user exit for inbound, to create hash value • 358

Security user exit

- example • 330
- function to register • 656
- inbound • 655
- outbound • 662
- overview of AUTACK example • 331
- specifying for acknowledgment profile • 233, 537
- specifying for input stream • 225
- specifying for output stream • 227

Segment

- adding to a document • 570
- adding to a standard version • 626
- adding to a wrapper • 669
- area as part of element name • 698
- area in document • 563
- composites or simple elements, repeating • 629
- conditions • 125, 637
- conditions, adding or modifying • 633
- conditions, deleting • 634
- configuration window • 616
- deleting • 627, 628
- deleting from a document • 571
- deleting from a wrapper • 670
- Edibasic validate routines • 124, 136
- fixed-length data, padded • 636
- level in document • 563
- list of documents or wrappers where used • 125, 130, 139, 639
- modifying • 571, 627
- modifying in a wrapper • 670
- occurrence as part of element name • 698
- occurrences, controlling with Edibasic • 584
- occurrences, Edibasic Do-Loop statement • 719
- occurrences, Edibasic For Each statement • 720

- occurrences, Edibasic For statement • 730
- occurrences, Edibasic method to initialize • 715
- occurrences, Edibasic method to process • 713
- occurrences, Edibasic method to reset • 715
- occurrences, Edibasic While statement • 728
- pre-processing user exit to add • 330
- properties, defining • 125, 139
- requirement in document • 568
- saving • 627
- sequence number as part of element name • 698
- Show Fields command • 616, 625
- Show Repeats command • 616
- tag as part of element name • 698
- tag delimiter character • 684
- tag, specifying for fixed-length segment • 616, 624, 630
- terminator character • 684
- versus loop, defining • 563
- Segment tag
 - defining • 564
 - specifying in fixed-length segment • 136
- Segments
 - characters to ignore between • 649
 - displaying all sequentially • 628
 - document, defining • 124
 - IO mode to mark end • 130
 - validate routines, adding • 631
 - validate routines, adding to elements • 631
 - validate routines, declaring variables • 632
 - validate routines, declaring variables on elements • 632
 - validate routines, printing • 586
 - variable-length, configuring in fixed-length standards • 153
 - wrapper elements, assigning to internal fields • 136
 - wrapper, defining • 136
- Select Case statement • 726
- Select Environment command • 525
- Sequence
 - component element in composite • 539
 - segment or loop in document • 564
- Service characters
 - alternative, for backward acknowledgment • 276, 287
 - alternative, for output • 273, 284
 - value and offset • 683
 - value and offset, specifying • 130, 676
- SetField function • 751
- Shortcuts
 - Test menu commands • 762

Workbench menu commands • 487

Signature

digital, configurations for user exit generation

• 357

digital, configurations for user exit validation

• 344

digital, user exit to generate • 347

digital, user exit to validate • 363

Source

location for inbound data on processing report • 47

location for output on processing report • 70

Source code for user exit examples • 330

Source location • 47

Space\$ function • 753

Spaces

adding to string • 753

leading and trailing, removing from string • 756

leading, removing from string • 744

post-processing user exit to remove • 330

trailing, removing from string • 751

Standard

inbound, on processing report • 47

proprietary, creating • 108, 111, 116

proprietary, defined • 116

Standard ID definition

acknowledgment, specifying trade agreement • 317

copying • 492

create with Partner Wizard • 315, 316

implications for processing control • 21

inbound, null wrapper • 149

list of acknowledgment profiles • 57

processing control • 257

purposes • 297

specifying a trade agreement for • 859

tasks for processing control and defaults • 307

tasks for standard identification • 300

trade requirement criteria • 23

Standard identification

by matching data fields • 858

by source location • 857, 861

default matching criteria, adding to wrapper • 676

definitions and tasks • 300

description of TRM task • 47

mandatory TRM task • 45

process • 836

using source location • 47

using wrapper content • 47

Standard types, See Data types • 119

Standard version

adding • 645

BCD data types, defining • 647

configuration window • 641

data types • 641

data types, defining • 646

data types, modifying • 647

defining • 118

deleting • 648

properties, defining • 120

saving • 648

Standata types, defining • 646

validate element codes • 649

Standards

adding • 645

defining • 115

identification of incoming • 17

loading public • 507

overview • 115

tasks to define • 117

using in a map • 107

Start method • 715

Stations • 47

Status

assignment of • 238

inbound document, on processing report • 70, 74

inbound wrapper, on processing report • 74

internal values • 237

processing, meaning of values on reports • 509

processing, option to report • 508

reporting on processing report • 512

Stop method • 715

Str\$ function • 754

StrComp function • 754

String

ASCII code value of first character • 729

case, changing to lower • 743

case, changing to upper • 756

characters, accessing leftmost • 743

characters, accessing middle • 745

characters, accessing rightmost • 750

characters, number of • 744

cross-reference look-up • 759

element data type • 698

example, converting from integer • 178

- format conversion from non-string date or number • 737
- literal • 698
- spaces, adding to • 753
- spaces, removing leading • 744
- spaces, removing leading and trailing • 756
- spaces, removing trailing • 751
- value of, convert from numeric expression • 754
- value of, convert to numeric value • 757
- value of, convert to numeric value and apply format • 757
- variable • 695
- variable, selecting part of • 724

String\$ function • 754

Subtraction operator • 707

Sum function • 755

Summary document

- outbound map, selected for trade agreement • 662
- outbound, configurations for AUTACK example • 354
- security user exit • 331
- specifying for acknowledgment profile • 233, 537
- specifying for trade agreement • 227
- user exit for, to create hash value • 346

Support

- related documentation • 6
- web site • 6

SWIFT

- document • 123
- element • 127, 141
- segment • 125, 139
- standard version • 120
- syntax support for document • 578
- syntax support for element • 596
- syntax support for segments • 636
- syntax support for standard version • 650
- syntax support for wrapper • 680
- wrapper • 130

syntax

- Edibasic functions • 728
- Edibasic methods • 712
- Edibasic operators • 706
- Edibasic statements • 717
- Edibasic, checking for errors • 585
- element names • 698
- variable names • 696

System date and time • 748

T

Tag

- element in a composite • 539

element or composite element in a segment • 623

segment or loop ID • 564

XML • 636

Tasks

configurations, basic • 16

content, defining • 121

envelope, defining • 128

icons for Test menu • 762

icons for Workbench menus • 487

list of TRM processing • 46

mandatory processing • 29, 45

map, creating • 167

map, creating for document • 171

map, creating for wrapper • 168

standard and version, defining • 118

standards, defining • 117

toolbar buttons, Test Setup • 761

toolbar buttons, Workbench • 487

Temporary variable

declaring • 696

description • 696

Tests

adding line breaks to output • 763

configuration text files, generating • 389

definitions, exporting • 403

deleting output automatically • 763

generating text files • 499

information required • 387

integration with production • 15

option to change viewing format • 389

option to delete output files • 389

options • 388

procedures for running • 390

processing report options • 388

saving definitions to a file • 773

setting flag in wrapper • 273, 284

switching between database environments • 529

Test window • 40

toolbar buttons • 761

Text files

acknowledgment profiles, generating • 233

configuration, generating • 389

document map, generating • 182

document, generating • 127, 572

generating for testing • 499

map, generating • 609

Standard ID, generating • 311

trade agreement profiles, generating • 228

- wrapper map, generating • 171
- wrapper, generating • 142, 671
- TimeSerial function • 755
- Toolbar buttons
 - Test • 761
 - Workbench • 487
- Tools menu commands • 490
- Trade agreement profiles
 - alias, specifying • 227
 - configuration • 20
 - configuration window • 652
 - decision matrix for matching • 61
 - default, on Standard ID • 310
 - defining • 222
 - determining where to attach • 22
 - error actions • 656
 - finding, description of TRM task • 61
 - identified on processing report • 61, 74
 - inbound acknowledgment, options required for reconciliation • 223, 239
 - inbound acknowledgment, without mapping • 239
 - inbound document and security options • 655
 - inbound document, specifying • 225
 - matching fields • 222, 653
 - matching fields for TRM task • 61
 - matching, mandatory TRM task • 45
 - null wrapper, defining • 148
 - options, specifying • 223
 - output properties, configuration window • 660
 - output properties, maps • 660
 - output properties, number to group outputs • 660
 - output properties, security user exit for summary document • 662
 - output properties, summary document • 662
 - outputs, list of • 658
 - outputs, specifying • 226
 - processing actions • 656
 - processing control • 22
 - random search from profiles list • 61
 - report, printing • 229
 - security document, specifying • 225
 - security user exit, specifying for input stream • 225
 - security user exit, specifying for output stream • 227
 - specifying as a default • 859
 - specifying for a partner • 779
 - specifying for a partner relationship • 825
 - summary document, specifying • 227
 - text files, generating • 228

- Trade agreements • 219
- Trading requirements, determining definitions • 11, 23
- Transaction set, See Document • 122
- Translation
 - date and time stamp on report • 47
 - processing option • 31, 36
 - reports, viewing example test • 43
 - routing output • 91
 - test, exporting definitions • 403
 - validation of wrappers and contents • 68
 - XML examples • 368
- Translation report, See Processing report • 388
- Translator Runtime Module
 - errors, list of • 425
 - mandatory tasks • 45
 - monitoring processing with audit user exit • 330
 - runtime exceptions, troubleshooting • 423
 - version and build on processing report • 47
- Trim\$ function • 756
- TRM • 45
- Troubleshooting
 - missing segments or elements • 159
 - parsing fixed-length data • 474
 - processing report • 469
 - text file generation errors • 483
- U**
 - UCase\$ function • 756
 - User exit
 - audit, example • 330
 - control reference, configuration requirement • 330
 - digital signature, example • 337
 - Edibasic, example • 330
 - Edibasic, to generate the digital signature • 347
 - Edibasic, to validate the digital signature • 363
 - function to call registered • 756
 - hash value, example • 337
 - post-processing, example • 330
 - post-processing, specifying • 130, 678
 - pre-processing example • 330
 - pre-processing, specifying • 130, 677
 - pre-processing, when called • 49
 - security, configurations for AUTACK example • 339, 354
 - security, example • 330
 - security, for summary document to calculate hash value • 346, 358
 - security, function to register • 656

security, specifying for acknowledgment profile • 233, 537

security, specifying for input stream • 225

security, specifying for output steam • 227

writing, considerations for AUTACK example • 339

User exits

adding to or removing from environment list • 521

examples • 9, 330

inbound data manipulation • 324

limits for an environment • 520

list of, for an environment • 520

overview • 323

projects, examples of Visual C++ and C++ Builder • 330

Register function • 329

restrictions • 337

types • 329

User function • 756

Utilities • 8

convert XML DTD of XSD to Edikit TRN • 369

V

Val function • 757

Validate Element Codes • 594

switch for document • 579

switch for standard version • 650

switch for wrapper • 680

Validate method • 716

how to use during parsing • 325

inbound data definitions • 324

page on Edibasic window • 585

results during parsing versus mapping • 325

Validate routines • 323

adding to elements in segments • 631

component elements, adding • 546

component elements, declaring variables • 547

composite elements, adding • 548

composite elements, declaring variables • 547

Document window • 122

documents, adding • 573

documents, declaring variables • 573

documents, declaring variables on segments • 574

Element window • 126, 140

elements within segments, declaring variables • 632

elements, adding • 592

elements, declaring variables • 593

Segment window • 124, 136

segments within documents, adding • 572

segments within wrappers, adding • 672

segments, adding • 631

- segments, declaring variables • 632
- viewing • 122
- Wrapper window • 129
- wrappers, adding • 671
- wrappers, declaring variables • 672
- wrappers, declaring variables on segments • 673
- ValidateAll method • 717
- Validation • 323
 - Edibasic code flag on component element in segment • 540
 - Edibasic code flag on element in segment • 624
 - Edibasic code flag on segment or loop in document • 564
 - Edibasic code flag on segment or loop in wrapper • 666
 - Edibasic user exit • 327
 - element codes, automatic • 594
 - inbound partner IDs, description of TRM task • 53
 - inbound recipient partner ID only • 33
 - inbound sender and recipient partner IDs • 36
 - inbound wrappers and contents, ERM task • 68
 - outbound data • 328
 - processing option • 31
 - routing output • 86, 91
- Value • 695
 - average, of elements for all occurrences within scope • 729
 - maximum, of elements for all occurrences within scope • 744
 - minimum, of elements for all occurrences within scope • 745
 - sum, of elements for all occurrences within scope • 755
- Value function • 757
- Variable-length segments • 153
- Variables • 693
 - data type, determining • 758
 - data type, determining if numeric • 742
 - data types • 695
 - declaring for document or wrapper • 613
 - declaring for loop, segment or composite element • 614
 - element, declaring • 615
 - example, local • 214
 - global • 165
 - global, example for user exit • 344, 347, 357, 363
 - global, statement to declare • 721
 - initialization • 157, 583
 - lifetime and scope • 697
 - limit on number • 696

- local, statement to declare • 718
- mapping example • 175
- scope • 582
- string, selecting part of • 724
- types and declaration • 696
- validate routines on component elements • 547
- validate routines on composite elements • 547
- validate routines on documents • 573
- validate routines on elements • 593
- validate routines on elements in segments • 632
- validate routines on segments • 632
- validate routines on segments in documents • 574
- validate routines on segments in wrappers • 673
- validate routines on wrappers • 672
- values, statement to assign • 724

Variant variable • 695

VarType function • 758

Version • 10

- standard, defined • 117
- TRM, on processing report • 47

Version, See also Standard version • 118

View menu commands • 489

Visual C++ user exits projects • 330

W

Weekday function • 759

While statement • 728

Windows menu commands • 491

Workflow • 15

Wrapper • 127

- acknowledgment flag • 273, 284
- adding • 667
- configuration window • 664
- default matching criteria, adding for standard identification • 676
- deleting • 668
- Edibasic validate routines • 129
- inbound, associating with mailbox • 17
- inbound, matching criteria • 18, 47, 304
- inbound, on processing report • 55
- inbound, parsing requirements • 50
- map, creating • 168
- map, generating text files • 171
- modifying • 668
- null, defined • 128
- null, defining • 17, 146
- null, example • 146

- outbound map, selected for trade agreement • 660
- outbound, specifying on acknowledgment profile • 231
- outbound, specifying on trade agreement • 227
- parse, description of TRM task • 50
- post-processing user exit, specifying • 678
- pre-processing user exit, specifying • 677
- properties, defining • 130
- properties, defining default matching criteria • 130
- report options • 664
- report, printing or previewing • 143, 674
- saving • 668
- segment, adding • 669
- segment, deleting • 670
- segment, modifying • 670
- service characters • 683
- service characters, specifying • 676
- Standard ID for null • 149
- standard identification • 47
- test flag • 273, 284
- text file, generating • 142, 671
- trade agreement profile for null, defining • 148
- validate element codes • 679
- validate routines, adding • 671
- validate routines, adding to segments • 672
- validate routines, declaring variables • 672
- validate routines, declaring variables on segments • 673
- Wrappers • 115
 - copying • 492
 - defined • 127
 - defining • 129
 - display all sequentially • 669
 - outbound, on processing report • 70
 - specifying default processing for inbound • 859
 - status of inbound, on processing report • 74
 - validate option • 31
 - validation of inbound, with contents as TRM task • 68
- Wrappers and content • 115, 223
 - service characters • 683
 - validation option • 33
- X**
 - X12 • 115, 506
 - data element reference number • 539, 588, 623
 - standards definitions supported • 7
 - translation example configurations, finding • 38

- translation example configurations, list of • 40
- translation example, testing • 41
- translation example, viewing test reports • 43
- translation tutorial • 9

XML • 367

- attribute • 596
- CDATA • 596
- converting to TRN • 374
- defining document • 373
- document • 123, 578
- document type definition, importing definitions from DTD • 369
- element • 127, 141, 596
- generating • 384
- mapping • 382
- schema, importing definitions from • 369
- segment • 125, 139
- standard version • 120, 650
- tags • 636
- translation examples • 368
- validation • 367
- wrapper • 130, 680

Xor operator • 710

Xref\$ function • 759

XrefR\$ function • 759

XSD, See XML data • 374

Y

Year function • 760