Etasoft
Building blocks of integration

Extreme Translator v1.9x User's Manual

Main website http://www.etasoft.com
Extreme Translator website http://www.xtranslator.com

Purpose

Extreme Translator was designed as lightweight Business-to-Business middleware integration suite. It was designed as extension and next generation product suite based on our successful Import/Export Studio software package. Primary purpose of this software suite is data translation and conversion from one format to another. We have incorporated both file-to/from-file, file-to/from-database translation. Our goal is to keep it simple and lightweight.

Requirements

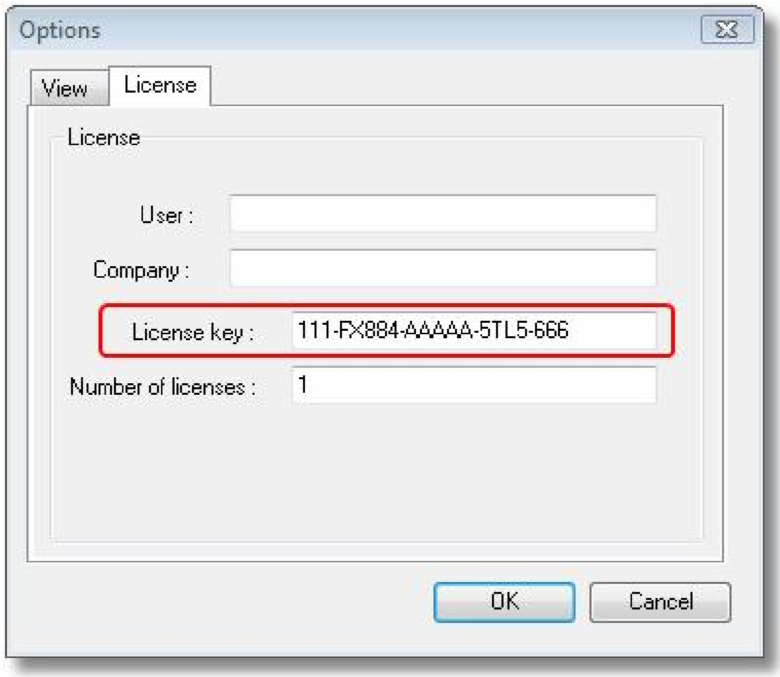
Table lists minimum hardware and software requirements for the application to run:

CPU	Pentium 3, 700 MHz
RAM	256 Mbt
HDD	2.5 Gbt
Other	Keyboard, mouse
Operating System	Windows XP/Vista/7, Server 2003

If you are working with big data files and map files that are more than 500Kbt in size you need much faster machine. This is recommended configuration:

CPU	Pentium 4, 1500 MHz
RAM	512 Mbt
HDD	2.5 Gbt
Other	Keyboard, mouse
Operating System	Windows XP/Vista/7, Server 2003

Enterprise or Lite Edition license



User :
Company :
License key : 111-FX884-AAAAA-5TL5-666
Number of licenses : 1

Evaluation version can be distributed free of charge. Licensing is based on number of installations. Retail version with setup license key can be installed on as many machines as many licenses have been purchased. It is possible to buy unlimited install and distribution license. You can find more licensing details on our website.
You can register and buy licenses from our website. After the purchase email will be send to you with the license key. You should go to submenu “Options” under menu “Project” and enter that key in “License Key” field.

If license key is not entered or has expired, translation will not run and license key has to be purchased.

Enterprise Edition is oriented towards companies and users that wish to automate data processing. Enterprise Edition can run as part of batch process, or be scheduled to run at certain times or time intervals via Extreme Processing process.
Lite Edition is for manual data processing only. It is much more restricted in runtime utilities that can be used to run maps. Lite Edition license does not permit map execution via command line.

The role of translators

Why are they needed? A translator is an application program designed to convert one electronic format into another and perform additional data conversion if desired. The industry term for applications designed to convert electronic formats is “translator” software.

Translator software can be acquired by
1. Developing a translator process in-house. Gives the maximum control over the design, quality, and functionality of the translator, but requires an investment in development, testing, and maintaining. *We provide integration and Developer SDK tools for this scenario.*

2. Purchasing or leasing vendor translator packages. It provides a tested application and a range of support services including upgrades and new releases as the standards evolve. *This is what our package is designed for.*

3. Contracting with a clearinghouse to perform the translator function. The clearinghouse offers a series of value-added services such as connectivity, a communications package, and trading partner interfaces, in addition to translation.

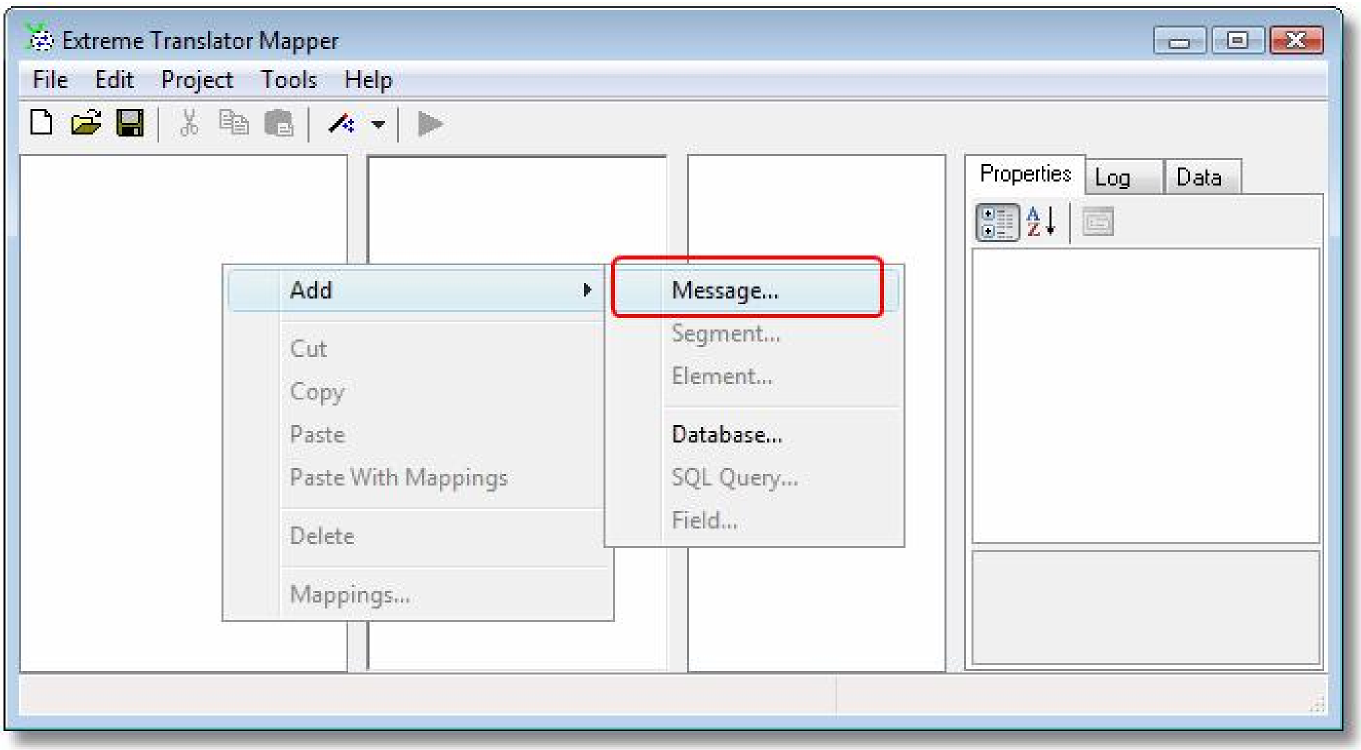
What translators can do?

Translators are designed to convert the data into transactions the receiving and sending system can recognize. They can perform the following functions:
1. Accept incoming standard formats and translate into other formats.
2. Strip and store data
3. Translate one file to many and many files to one file
4. Perform data validation
5. Reformat certain data items, example change date time formats, pad numbers with zero
6. Insert processed data into relational databases

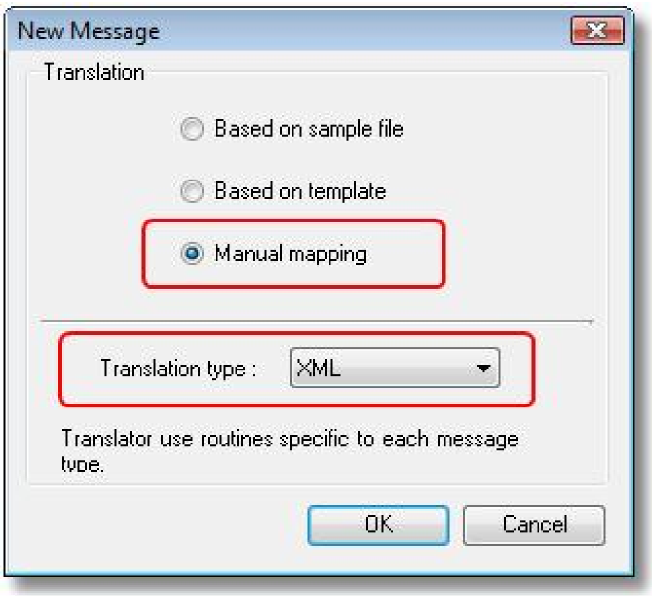
Map Editor

Map editor is divided into four panes. First and third pane is for adding new map items. Second pane is to display relations between map items, and fourth pane is the one used to display selected item properties.
There are many ways to add items to the map: use wizard, use menu “Add..”, use “Copy” and “Paste”, or use Drag-n-Drop. In any case you need to have at least one root item inserted at the top. This can be the item that represents your “Message” or “Database Connection”.
Translator supports one or many-to-one or many mappings. That means you may have one or many messages (files) mapped to one or many output messages. So you can have one or many root items on each side.
Simplest way to add some items is to use “Add…” menu item, however there are some limitations in order what can be added to what:

1. Only segments can be added to the message.
2. Elements and segments can be added to segments
3. Nothing can be added to elements. In other words: elements can not contain other items



Add message to the map.



Based on sample file

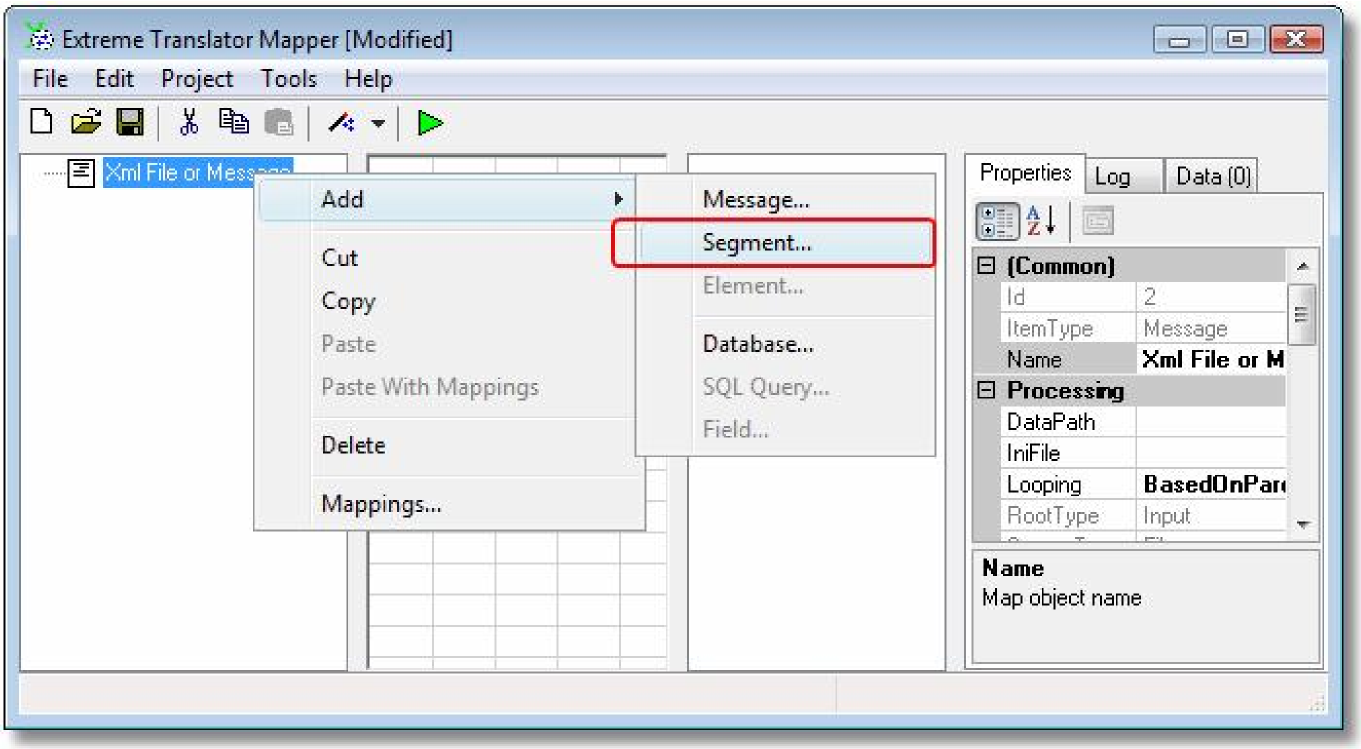
Based on template

[✓ Manual mapping]

Translation type : XML

Translator use routines specific to each message type.

Choose translation type that is most accurate to the message (file) you want to process. You can change translation type later, however map and map item properties are different depending on what translation type you are using. Example: in XML property StartTag means actual XML tag without “<” and “>” brackets, so in XML translation StartTag = “mytag”, if you are planning to parse XML file as simple text file you should set StartTag = “<mytag>”, just because simple text parsing does not know what XML tags are and whole XML data is just plain text with some separators. For EDI segments you do not need to have EndTag or LoopEndTag defined, because translator takes those properties from Message SegmentSep property.



Add first segment under the message.

Repeating this task you can define whole map. It is tedious and time consuming. It is much faster to use wizard and have map ready for you in minutes. However there are times when manual mapping is the only option because input or output data is so complex that no generic rules can be applied to create the map from scratch.

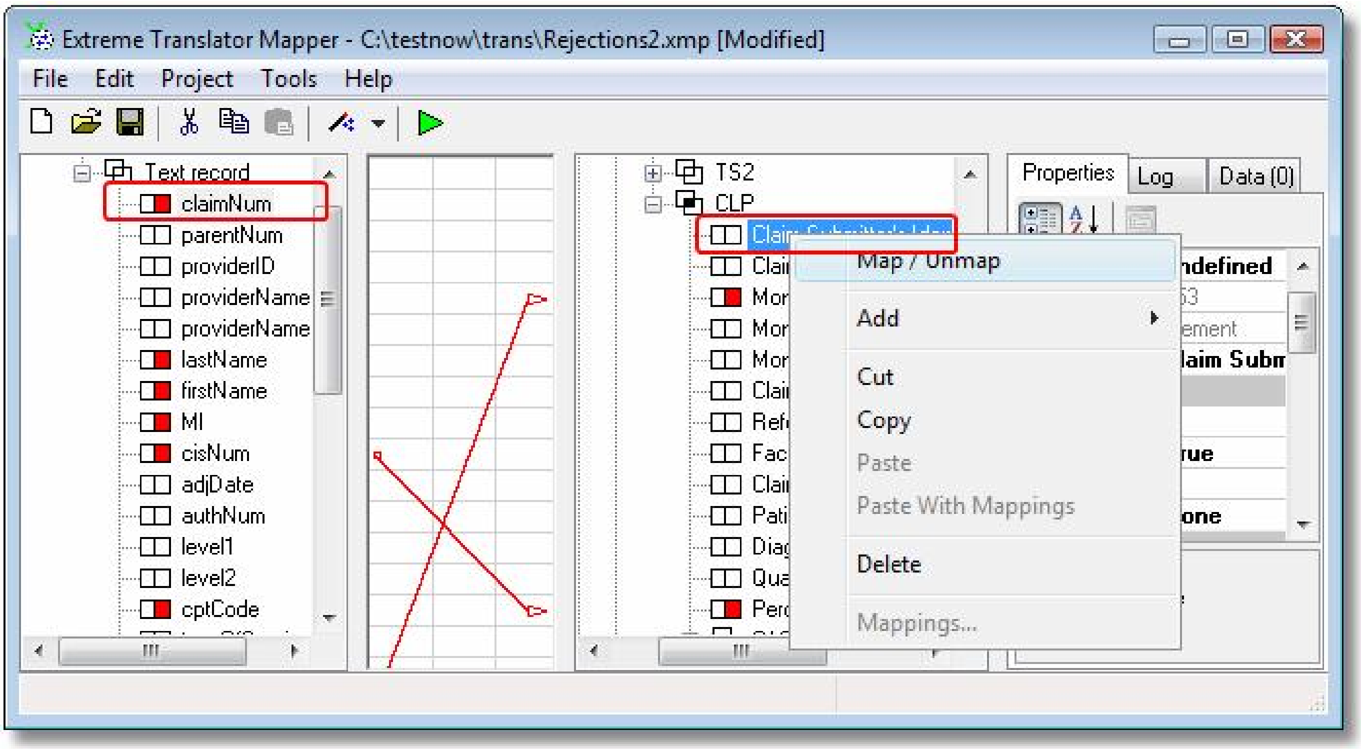
You can also use Copy, Cut and Paste to move items around. All these actions can be applied both to first and third panes. So, you can copy items from one tree view and paste into the second.

Drag and drop using mouse can be performed as well, however you should be careful using it because it is very easy to drag-n-drop items and have them added to incorrect location. Sometimes it is even better to disable drag-n-drop. You can do it in under menu “Options”. If you hold “Ctrl” key pressed during drag-n-drop operation it will result into copied and pasted items in the new location. If “Ctrl” key is not pressed it will result into cut and paste operation.

Note: You can map/un map items that are at least two levels down in the tree view. If it is root item or item directly attached to the root (nested only one level) Map/Un map menu option will be disabled. The reason is that you should define loops, SQL queries and segments under root, not real data items that can be mapped. In case if you do not have any loops, just define at least one segment under the root and hang all the items under it. That way you can map them all and segment is just a placeholder.

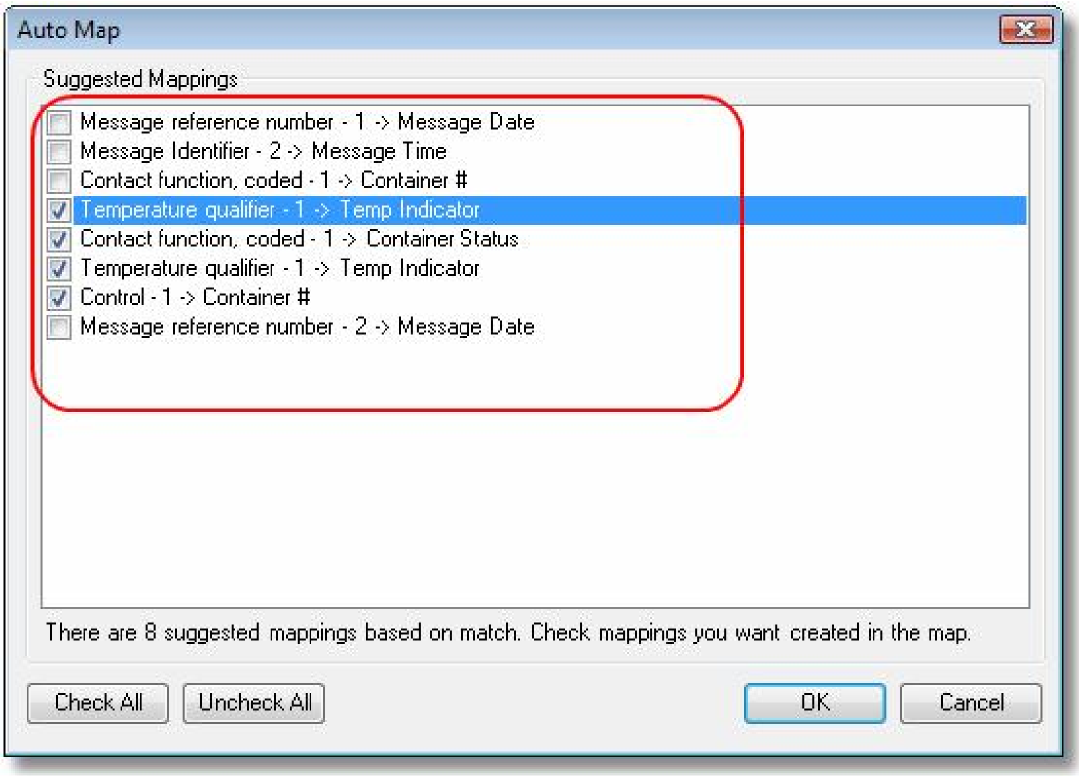
How to do the mapping?

After you define both map segments and elements for input and output you can use popup menu in the right tree view. Follow these steps:
1. Click on the item on the left tree view
2. Click on the item on the right tree view
3. Press right button on the mouse to get popup menu
4. Click menu item "Map/Unmap".



Perform mapping using right popup menu Map/Unmap.

There is also a way to create mappings faster using Auto Map feature. It is available via “Edit” menu. It tries to suggest mappings based on translation type and matching item names on both sides of the map.



- [ ] Message reference number - 1 -> Message Date
- [ ] Message Identifier - 2 -> Message Time
- [ ] Contact function, coded - 1 -> Container #
- [x] Temperature qualifier - 1 -> Temp Indicator
- [x] Contact function, coded - 1 -> Container Status
- [x] Temperature qualifier - 1 -> Temp Indicator
- [x] Control - 1 -> Container #
- [ ] Message reference number - 2 -> Message Date

There are 8 suggested mappings based on match. Check mappings you want created in the map.

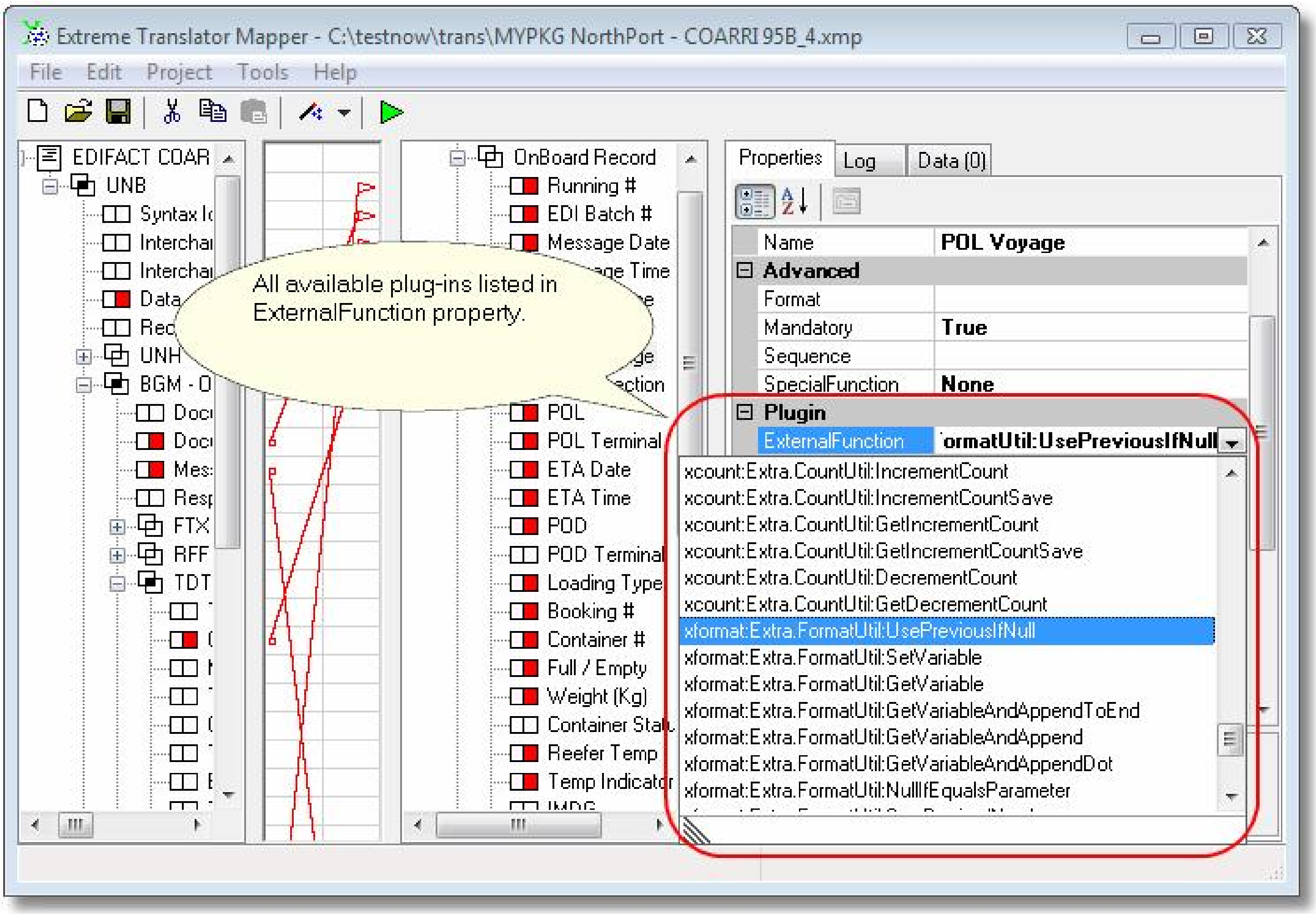
Once mappings are checked and accepted, they are added to the map.

Integration

There are number of ways for integrating middleware products. We try to support most of options available. Integrations supported: Batch Mode processing, stand-alone Extreme Processing Script Runner, Developer SDK for .NET, ActiveX component.

Plug-ins and .NET components

Translator engine can be integrated into application by using .NET components and classes exported from the translator library. That way translator maps can be executed directly from the application without a need to run external processes using shell commands. In order to receive translator Software Development Kit you need to obtain Developer license.


Extreme Translator Mapper - C:\testnow\trans\MYPKG NorthPort - COARRI 95B_4.xmp
File Edit Project Tools Help
Properties Log Data (0)

All available plug-ins listed in ExternalFunction property.

Name	POL Voyage
⊟ Advanced	
Format	
Mandatory	True
Sequence	
SpecialFunction	None
⊟ Plugin	
ExternalFunction	ormatUtil:UsePreviousIfNull

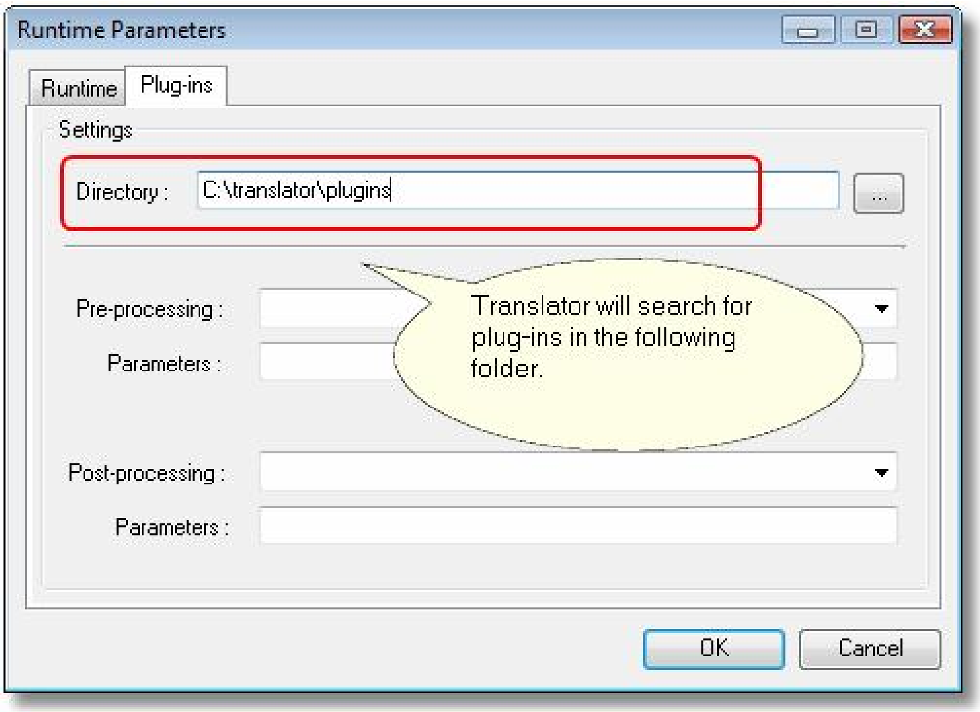
```
xcount:Extra.CountUtil:IncrementCount
xcount:Extra.CountUtil:IncrementCountSave
xcount:Extra.CountUtil:GetIncrementCount
xcount:Extra.CountUtil:GetIncrementCountSave
xcount:Extra.CountUtil:DecrementCount
xcount:Extra.CountUtil:GetDecrementCount
xformat:Extra.FormatUtil:UsePreviousIfNull
xformat:Extra.FormatUtil:SetVariable
xformat:Extra.FormatUtil:GetVariable
xformat:Extra.FormatUtil:GetVariableAndAppendToEnd
xformat:Extra.FormatUtil:GetVariableAndAppend
xformat:Extra.FormatUtil:GetVariableAndAppendDot
xformat:Extra.FormatUtil:NullIfEqualsParameter
```

List of available functions stored in libraries in plug-in directory.

However you can use plug-in feature and add your own functionality to the translation. You do not need any special license to use and develop plug-ins. Special functions such as specific date time conversions, formatting can be achieved using external .NET libraries. Each element and segment in the map can have attached to external functions. External functions must have certain interface to be integrated into translation:
1. Should be compiled into .NET class library (DLL) and placed into /plugins directory.
2. Class must have default constructor that has no parameters.
3. External function method should have special signature: return string, and take three parameters – integer map object Id, actual processing data as string, and function parameters coded in the map as string as well.

Library can be built using VB, C++ or C#. /pluginsamples directory contains samples of plug-in functions in C#. You are free to copy the code and modify it for your own needs. Library can be built using IDE tools or build from command line. If you use command line tools, go to the /pluginsamples directory and in order to make a build type:
csc.exe /t:library EDIMain.cs /nologo

Metadata about plug-ins is loaded into translator map editor only once during startup. If you update or drop new plug-ins library into plug-ins directory map editor will not list all the new external functions until you close and reopen editor again.


Runtime Parameters
Runtime
Plug-ins
Settings
Translator will search for plug-ins in the following folder.
OK
Cancel

Directory : C:\translator\plugins
Pre-processing :
Parameters :
Post-processing :
Parameters :

External functions are stored in subdirectory called “plugins” by default.

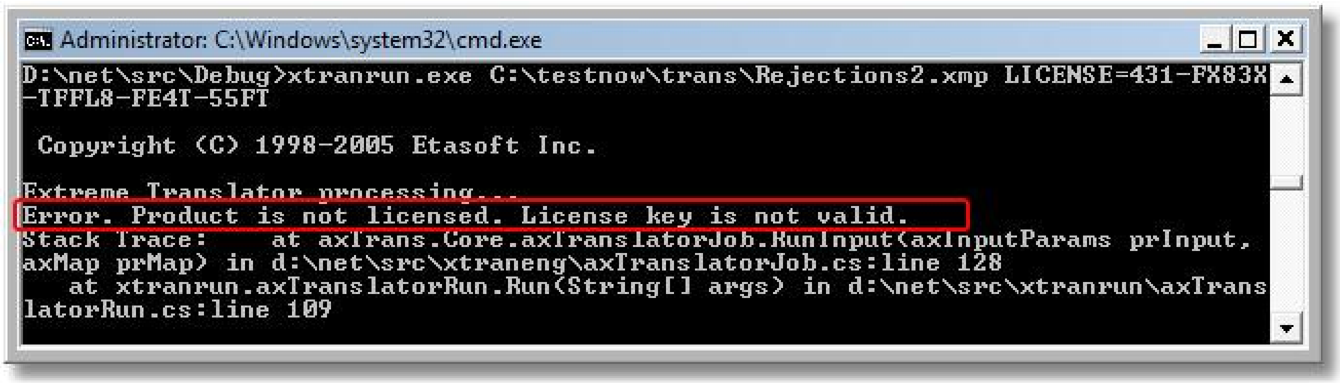
Plug-in can be passed into translator batch process via parameter PluginDirectory. Example: PluginDirectory=C:\mytransalator\myplugins. In this case it will overwrite value used in the map “Runtime Parameters” dialog.

Pre-processing plug-in is called at the beginning of processing, and post-processing plug-in is called at the very end of translator processing. Any actions can be performed in these plug-ins: it can be file deletion, renaming or even file sending by email operations. When pre-processing plug-in is called passed in Id equals 0, and when post-processing plug-in is called passed in Id equals 1. If pre-processing plug-in returns **null**, it is treated as translation abort event and translation does not continue. This feature can be used to receive data via some network communication, and if data is not present, do not run actual translation.

Command line batch processing

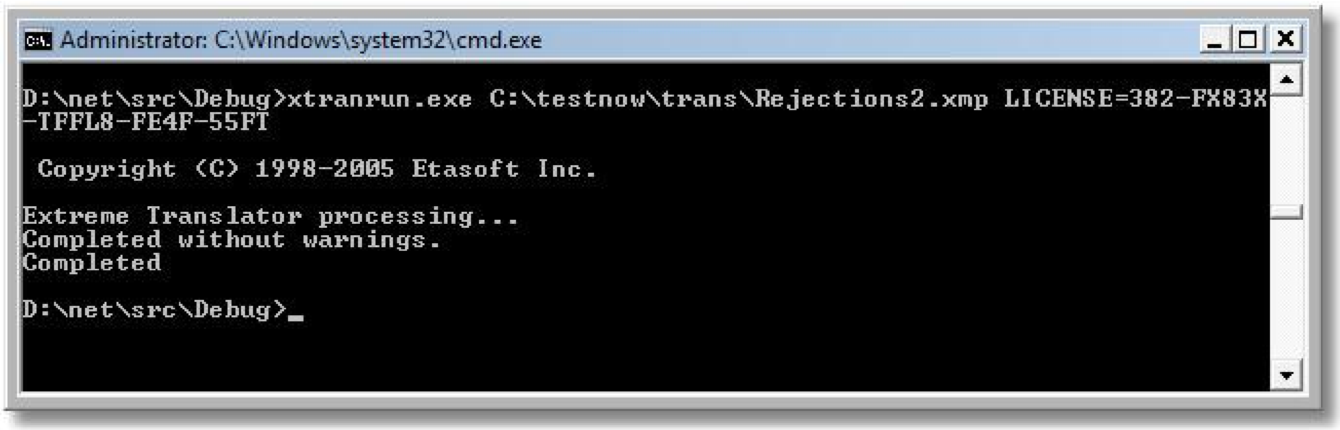
In order to run translator in Batch Mode you need to use utility called "xtranrun.exe". It should be located in main application installation directory. If you want to see the usage simply run it without any parameters in command line.
Batch utility returns 0 (zero) in case if processing has completed successfully and –1 (minus one) if it has failed.

Usage is
xtranrun.exe map_file_name.xmp License=YOUR_LICENSE [optional_datapath1] [optional_datapath2]



```
D:\net\src\Debug>xtranrun.exe C:\testnow\trans\Rejections2.xmp LICENSE=431-FX83X
-TFFL8-FE4T-55FT

 Copyright (C) 1998-2005 Etasoft Inc.

Extreme Translator processing...
Error. Product is not licensed. License key is not valid.
Stack Trace:    at axTrans.Core.axTranslatorJob.RunInput(axInputParams prInput,
axMap prMap) in d:\net\src\xtraneng\axTranslatorJob.cs:line 128
   at xtranrun.axTranslatorRun.Run(String[] args) in d:\net\src\xtranrun\axTrans
latorRun.cs:line 109
```

Running the map in command line with invalid or expired license key produces error.



```
D:\net\src\Debug>xtranrun.exe C:\testnow\trans\Rejections2.xmp LICENSE=382-FX83X
-IFFL8-FE4F-55FT

 Copyright (C) 1998-2005 Etasoft Inc.

Extreme Translator processing...
Completed without warnings.
Completed

D:\net\src\Debug>_
```

Running the map with fixed input and output files.

Example 1:
You want to perform file translation, and you have made a map for it. If you pass Input and Output parameters you can process specific file each time. You can also use wildcards to process multiple input files.

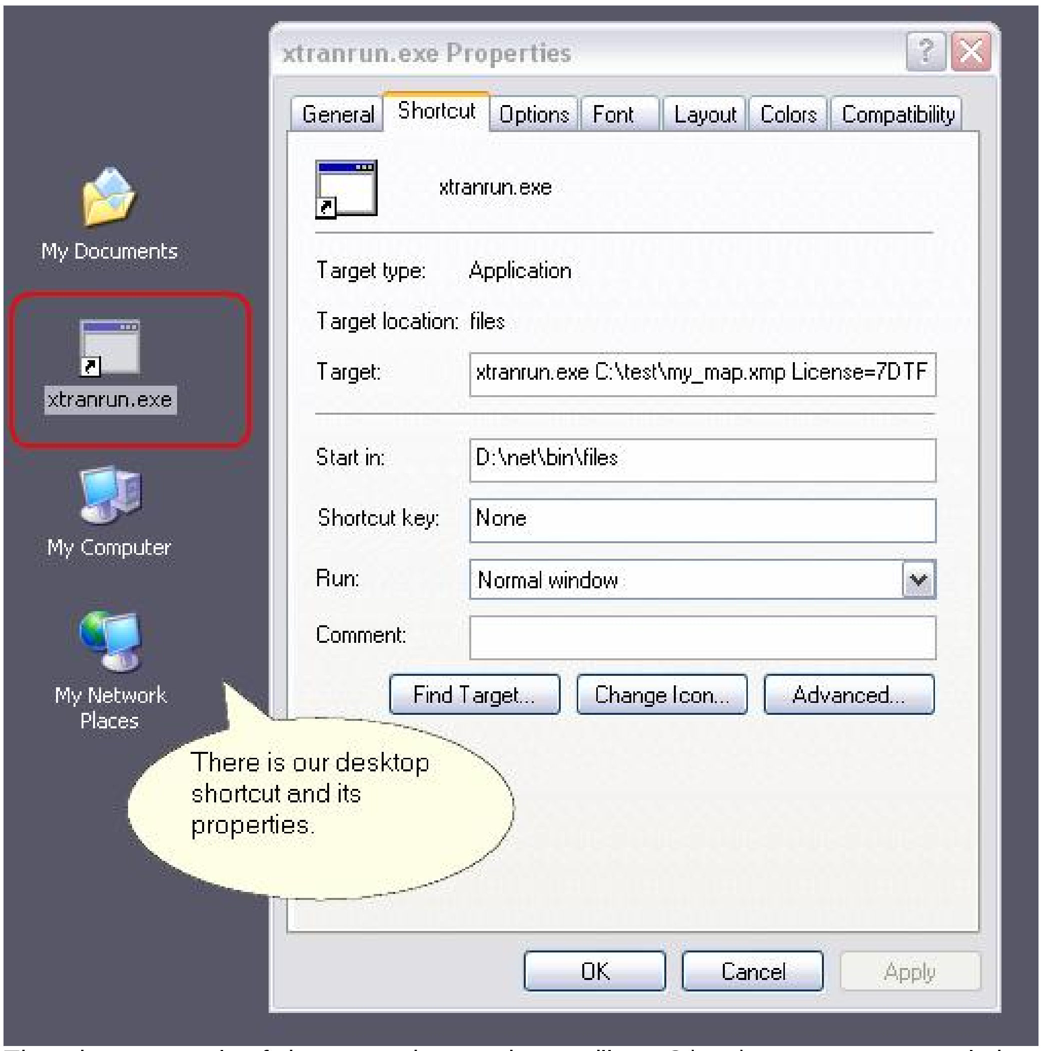
xtranrun.exe C:\test\my_map.xmp Input=C:\temp\data*.txt Output=C:\temp\output.txt License=YTRD3-34DFFZ

Example 2:
You do not want to use constants like in previous example. You want to hard code data path into the map for simplicity. In this case all you have to do is supply one parameter in command line and it is the map file name. PluginDirectory parameter lets change the folder name for the plug-ins.

xtranrun.exe C:\test\my_map.xmp License=7DTF-CDF74-DFA PluginDirectory=C:\myplugins

This assumes that you have put correct values into *DataPath* properties of the map and they are pointing to input and output files. Plug-in can be passed into translator batch process via parameter PluginDirectory. Example: PluginDirectory=C:\mytransalator\myplugins. In this case it will overwrite value used in the map "Runtime Parameters" dialog.

You can also setup shortcut on the desktop so every time it is double clicked, specific map would run.



xtranrun.exe

Target type: Application

Target location: files

Target: xtranrun.exe C:\test\my_map.xmp License=7DTF

Start in: D:\net\bin\files

Shortcut key: None

Run: Normal window

Comment:

There is an example of shortcut and properties. It will run C:\test\my_map.xmp translation whenever executed.

Passing parameters via command line

Most property values set in translator Mapper can be overwritten by passing new values to them via command line.
This is very powerful feature if you need to do some pre-processing before map is called and would like to change map properties during runtime on each map execution.
Parameter passing might look something like "Id:138.Mandatory=true". Here “Id:138” means the item that has property Id equal to 138 should have its property Mandatory set to “true” during map execution. Whole command line may look like:

```
xtranrun.exe C:\test\trans\mapping_new2.xmp "Id:138.Mandatory=true" License=T3D-6UXF3D-U6LXF
```

Two or more properties can be passed in as parameters, for example: Mandatory and Format.

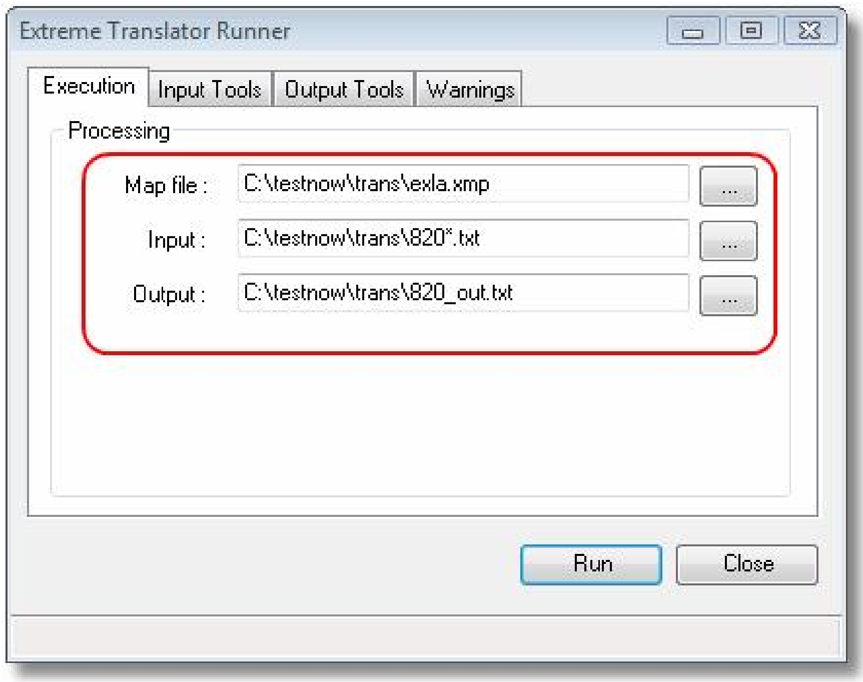
```
D:\net\trans\Debug>xtranrun C:\test\trans\mapping_new2.xmp "Id:138.Mandatory=true" "Id:138.Format==Value(TEST)"
License=T3D-6UXF3D-U6LXF
```

In this case Format property for item Id 138 is being set to =Value(TEST) string.

There are limitations to command line parameters that can be passed in to properties:
1. Item with that Id must exist in the map in order for parameter to be accepted.
2. Property must be not read-only. It can be changed via Mapper.
3. Property must be string, integer or Boolean value. Fixed dropdown list type properties cannot be reassigned or changed via command line.

Dialog based processing

Files can be processed using dialog screen when operator has to enter data and map file names. You may use “xtranrunw.exe” to do that type of processing. This utility accepts map file name as parameter. If you start it with map file name in shortcut parameters, “Map file:” field will be pre-populated.



Map file : C:\testnow\trans\exla.xmp
Input : C:\testnow\trans\820*.txt
Output : C:\testnow\trans\820_out.txt

Dialog based processing.

What are the maps? How to do the mapping?

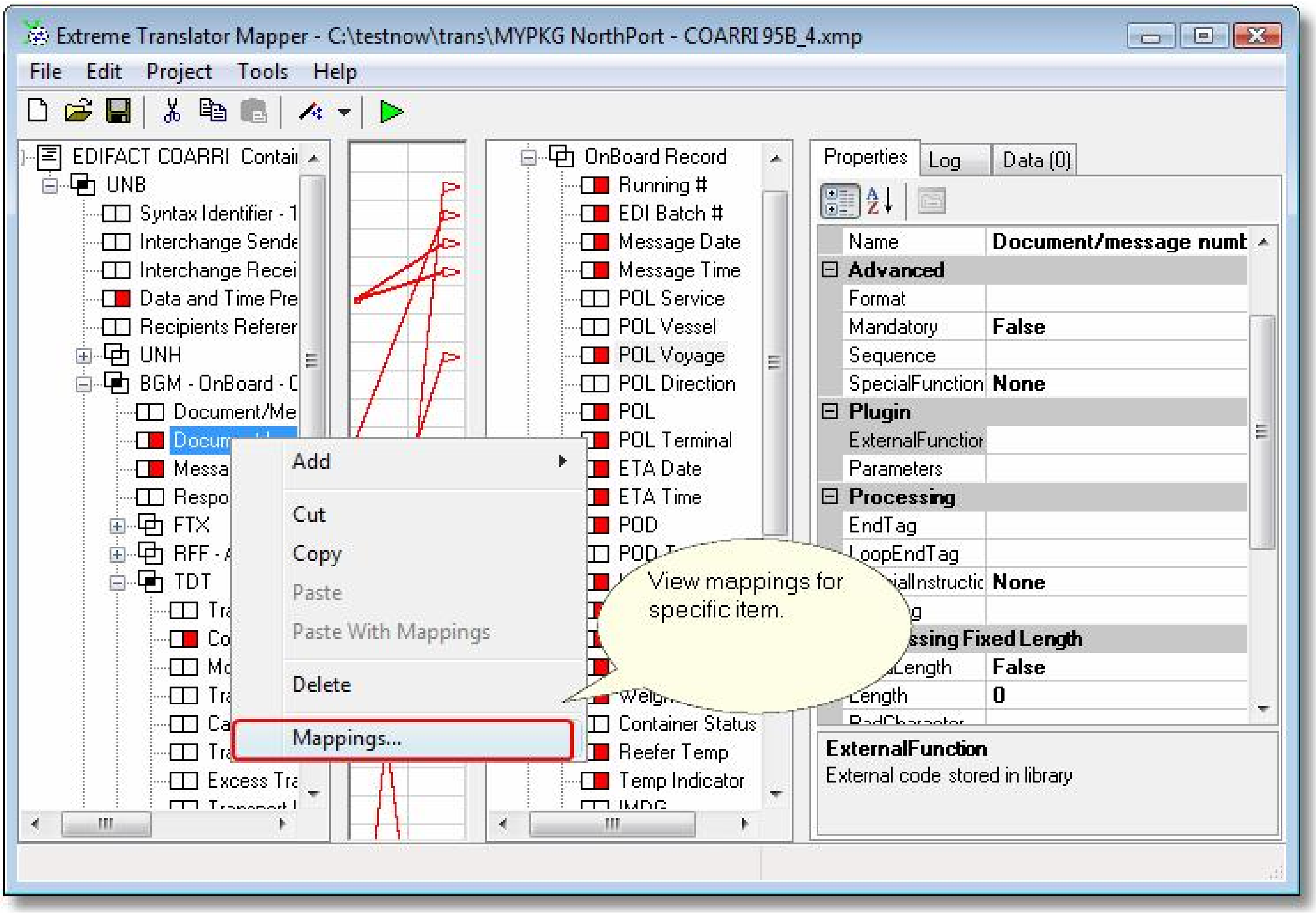
Translator maps are sets of data translation rules defined using graphical mapping tool and saved in *.xmp files. You can use those map files to process data and transform it into some other format or relational database. You do not have to write any code and you do not have to know any programming language to use the mapping tool for most of scenarios. The main idea is that you should be able to make mapping point-and-click and have it ready in an hour or two. However this is in ideal world, in reality sometimes very complex business rules should be applied and they just can not be simply defined by changing properties. In order to solve those problems we introduced scripting support into the product.

Also mapping can take a long time if you have never used other mapping tools before. If it is a new experience to you it may take longer than a few hours to make the mapping work. But this time will decrease tremendously after you get used to the tool.

Mapping process goes from left to right on the screen divided in two. You have to define at least one root item on both sides and one-or-many contained items underneath. Each root item represents a file or database connection. So it is the source or destination of data. You should define *DataPath* property in order for map to be functional. *DataPath* can be path to actual data file for input or output or it can be a constant value. In case if it is a constant value, you will be able to pass real path to the data via command line parameters. Please read more about it in a topic that describes how to pass data from command line.

However you can use wizards to do the mapping for you. In many cases they produce one side of the map, it can be input or output, and if you run wizard twice you can have most of the map created in minutes. All you will have to do is finish mapping using "Map/Unmap" menu in popup menu on the tree view in the right side.

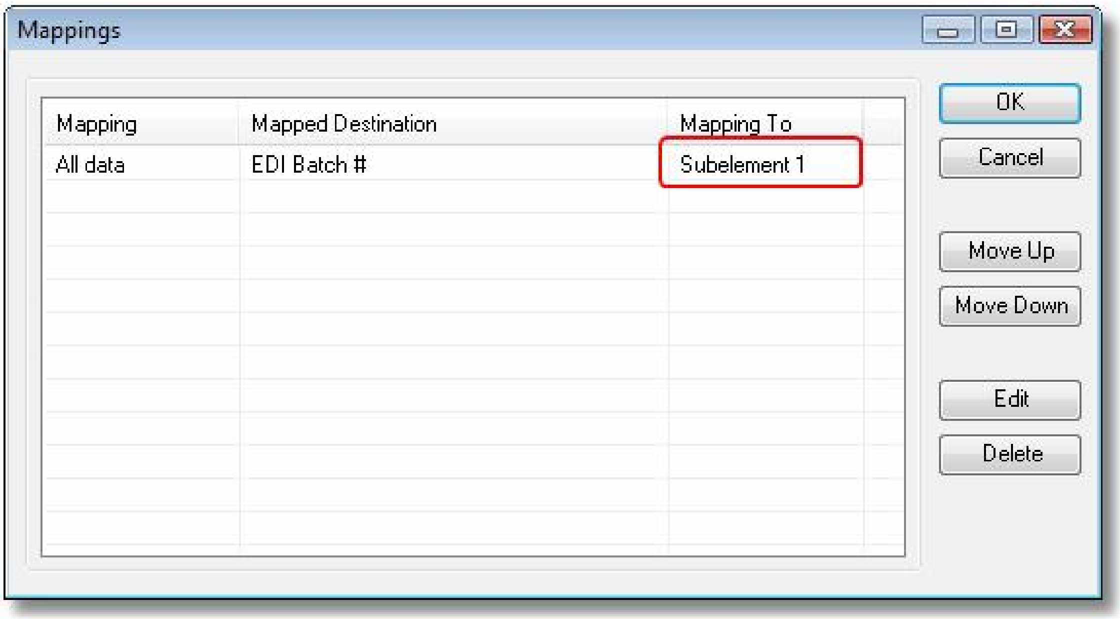
You can view mappings displayed as red arrow lines. However lines are painted only if mapped item is visible (expanded) on both tree views. You can also edit mappings by clicking “Mappings” menu item on the left side of the screen.


Extreme Translator Mapper - C:\testnow\trans\MYPKG NorthPort - COARRI 95B_4.xmp
File Edit Project Tools Help
Add
Cut
Copy
Paste
Paste With Mappings
Delete
Mappings...
View mappings for specific item.
Properties
Log
Data (0)
ExternalFunction
External code stored in library

Name	Document/message numb
⊟ Advanced	
Format	
Mandatory	False
Sequence	
SpecialFunction	None
⊟ Plugin	
ExternalFunction	
Parameters	
⊟ Processing	
EndTag	
LoopEndTag	

This is how to reach mappings for editing.

Mappings can be rearranged, deleted or edited. When you use menu option “Map/Unmap” all data from the item is mapped to the destination (output). But you can change this default mapping option and specify actual sub element you want to extract and use. This is especially useful for EDIFACT mappings where many data elements are composite of sub elements.



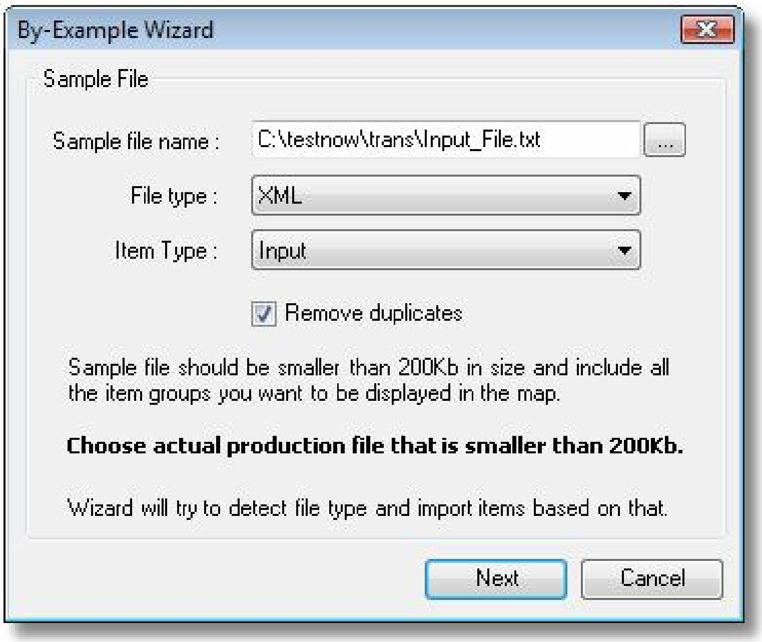
Mapping	Mapped Destination	Mapping To
All data	EDI Batch #	Subelement 1

There is how you can map to or from sub elements.

Wizards

There are two major wizards. One is based on template model and another one on sample data you can provide.

1. Add from template wizard is like storage of existing maps that are already pre-built for you. However those maps are mostly for standard implementations of EDI X12 or EDIFACT documents. You may also save some of your pre-built maps into templates for future use.
2. Add by example wizard is a bit more complicated. It is based on features of certain formats and somewhat intelligent in a way that it can read the file, detect file structure and create most of map objects based on example provided.



Sample file name : C:\testnow\trans\Input_File.txt

File type : XML

Item Type : Input

[✓] Remove duplicates

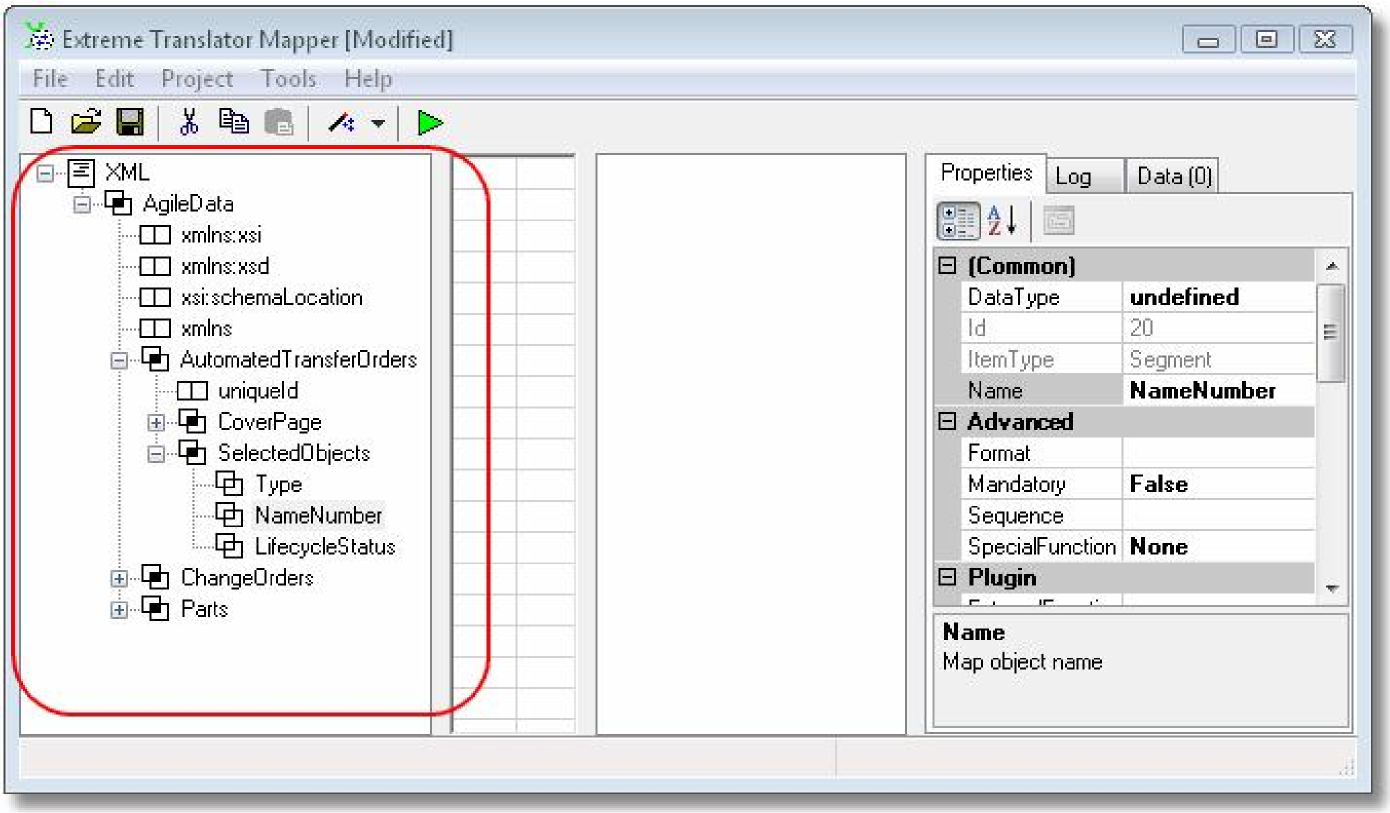
Sample file should be smaller than 200Kb in size and include all the item groups you want to be displayed in the map.

Choose actual production file that is smaller than 200Kb.

Wizard will try to detect file type and import items based on that.

By-Example wizard takes sample file and produces the map.

There are certain requirements to the files you can feed into this wizard:
a) File should have all the items you expect to have in real production environment. In most cases we recommend to use actual file from production because it should reflect actual data you are sending or receiving.
b) File should be small. Recommended size should be less than 20Kbt. If you try to use files that are more than 100Kbt in size wizard will take a long time to process it. It may even take more than 10 minutes.

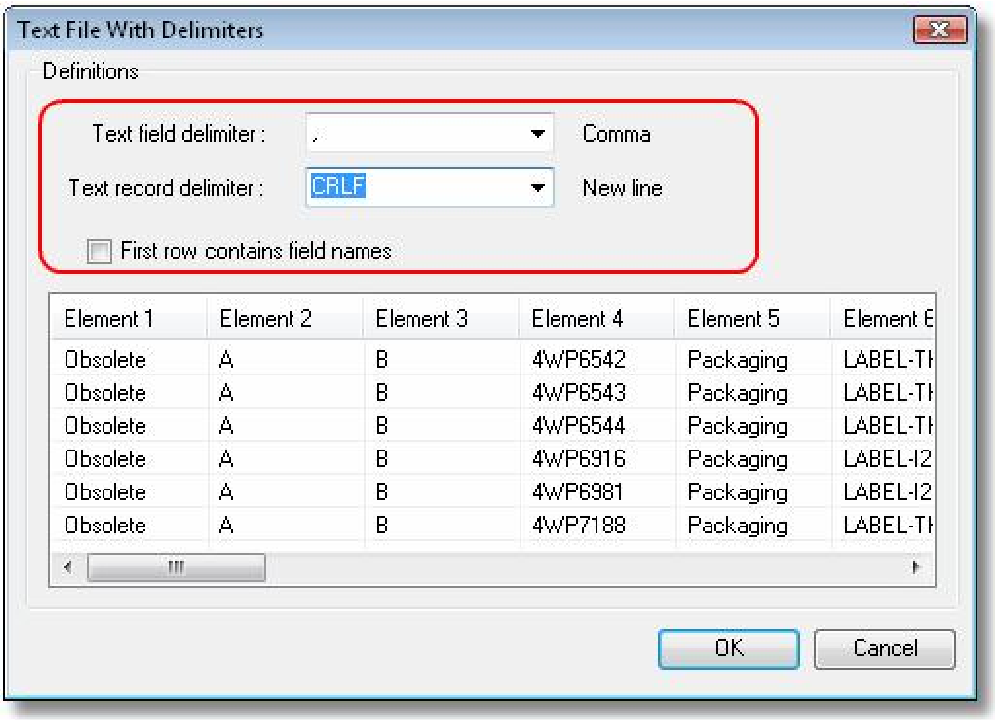

Extreme Translator Mapper [Modified]
File Edit Project Tools Help
Properties Log Data (0)

- XML
  - AgileData
    - xmlns:xsi
    - xmlns:xsd
    - xsi:schemaLocation
    - xmlns
    - AutomatedTransferOrders
      - uniqueId
      - CoverPage
      - SelectedObjects
        - Type
        - NameNumber
        - LifecycleStatus
    - ChangeOrders
    - Parts

(Common)	
DataType	**undefined**
Id	20
ItemType	Segment
Name	**NameNumber**
Advanced	
Format	
Mandatory	**False**
Sequence	
SpecialFunction	**None**
Plugin	

Name
Map object name

Imported sample XML file.



Text field delimiter : , Comma

Text record delimiter : CRLF New line

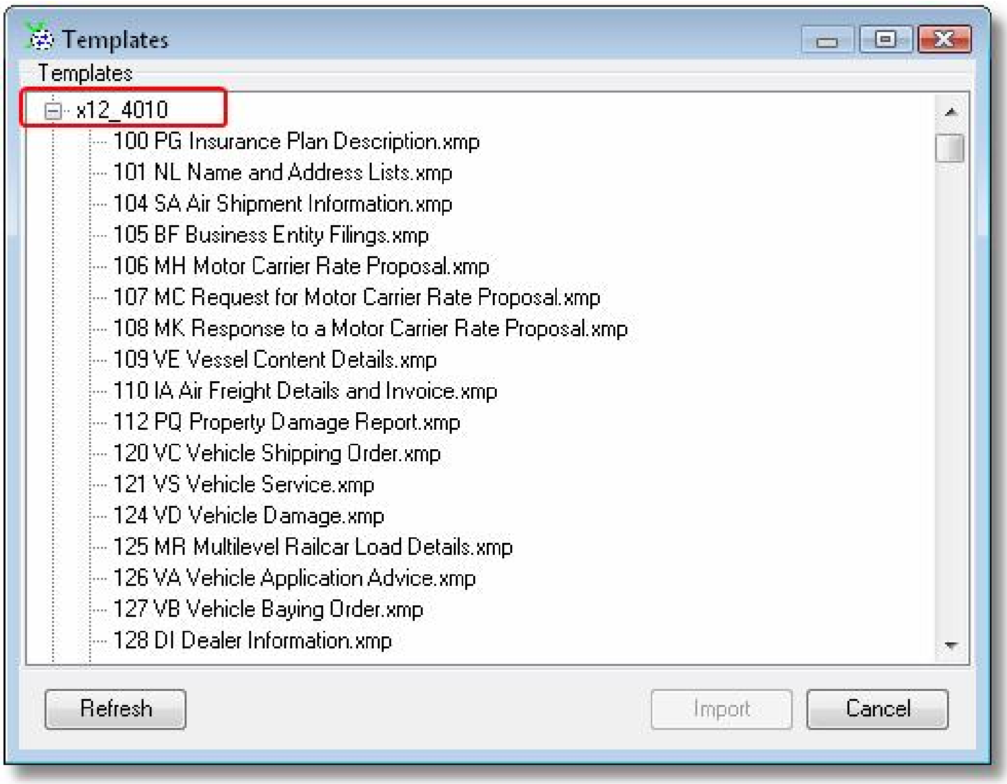
First row contains field names

Element 1	Element 2	Element 3	Element 4	Element 5	Element 6
Obsolete	A	B	4WP6542	Packaging	LABEL-TI
Obsolete	A	B	4WP6543	Packaging	LABEL-TI
Obsolete	A	B	4WP6544	Packaging	LABEL-TI
Obsolete	A	B	4WP6916	Packaging	LABEL-I2
Obsolete	A	B	4WP6981	Packaging	LABEL-I2
Obsolete	A	B	4WP7188	Packaging	LABEL-TI

If you choose “Text file with delimiters”, wizard will give you data preview based on delimiters you choose.

Templates

Software comes with over 5000 pre-built templates. If you need a template for some standard mapping that is not included in templates let us know and we will create template for you free of charge. If you develop a map and you think it can be useful for others using Extreme Translator you may send the map to us and we will consider it for inclusion into our next release of the product as a sample map or template. Special mapping services are also available check our website for details.

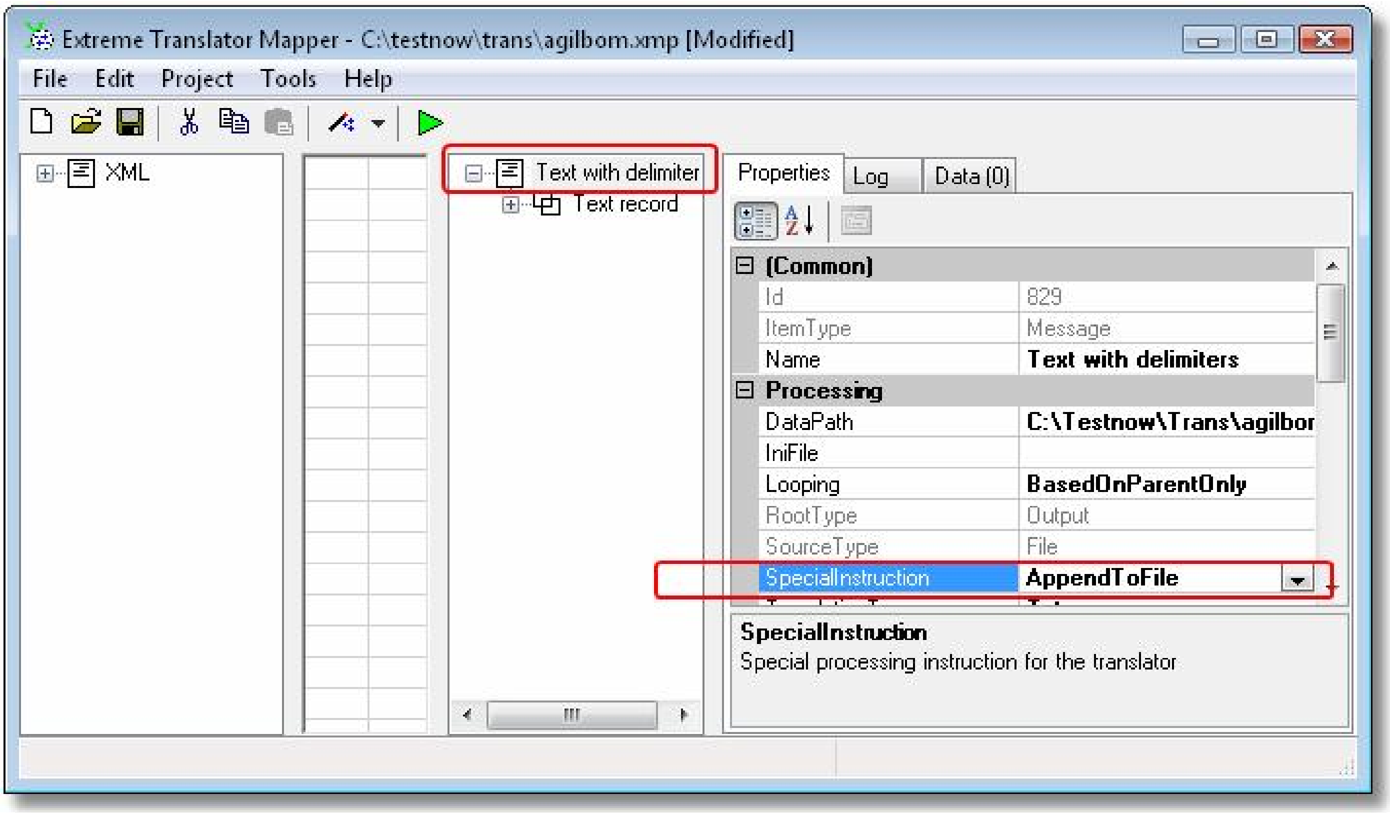


x12_4010
- 100 PG Insurance Plan Description.xmp
- 101 NL Name and Address Lists.xmp
- 104 SA Air Shipment Information.xmp
- 105 BF Business Entity Filings.xmp
- 106 MH Motor Carrier Rate Proposal.xmp
- 107 MC Request for Motor Carrier Rate Proposal.xmp
- 108 MK Response to a Motor Carrier Rate Proposal.xmp
- 109 VE Vessel Content Details.xmp
- 110 IA Air Freight Details and Invoice.xmp
- 112 PQ Property Damage Report.xmp
- 120 VC Vehicle Shipping Order.xmp
- 121 VS Vehicle Service.xmp
- 124 VD Vehicle Damage.xmp
- 125 MR Multilevel Railcar Load Details.xmp
- 126 VA Vehicle Application Advice.xmp
- 127 VB Vehicle Baying Order.xmp
- 128 DI Dealer Information.xmp

Import existing templates for most of standard messages of EDI X12, EDIFACT, SAP.

Translation

This is a simple and brief topic because we do not guarantee that translation logic will remain unchanged in next version of the software. However we try to minimize the impact and keep versions compatible.
Now translator uses different translation logic based on the "translation type" property defined for the root item in the map. EDI X12 and EDIFACT translation is based on segment splitting, XML translation is based on XML parser and Text based translation uses complex file parsing based on offsets and data blocks.
Translator overwrites output files during processing by default. You may change it to append using SpecialInstruction property on output root item in the map.


Extreme Translator Mapper - C:\testnow\trans\agilbom.xmp [Modified]

(Common)	
Id	829
ItemType	Message
Name	Text with delimiters
Processing	
DataPath	C:\Testnow\Trans\agilbor
IniFile	
Looping	BasedOnParentOnly
RootType	Output
SourceType	File
SpecialInstruction	AppendToFile

SpecialInstruction
Special processing instruction for the translator

Set this property on the output root item if you want data to be appended to the output file.

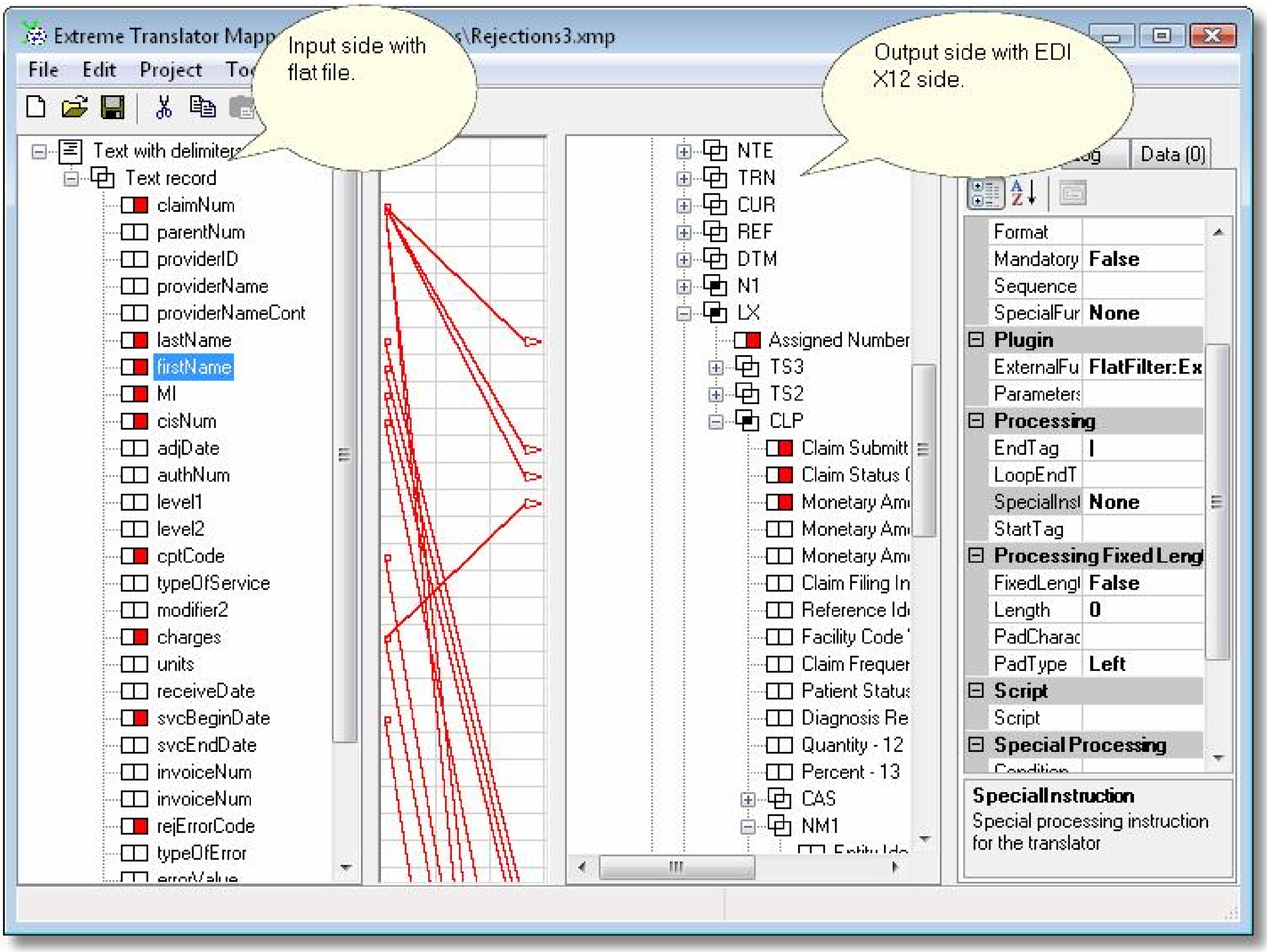
EDI X12, EDIFACT Translation

List of most important properties for EDI X12 and EDIFACT maps

StartTag	Used only for segments (loops). It should be setup to segment name, example: ISA, ST, SE, UNA, UNB, etc.
EndTag, LoopEndTag	Not used
Mandatory	Should be setup "true" for mandatory EDI or EDIFACT items, such as ISA, GS, ST, UNA, UNB, etc.
AutoDetect	If setup to "true" will try to locate segment and element separators
UseSep	If setup to "true" will use separators setup in other properties. It is opposite to "AutoDetect" property
SegmentSep, ElementSep, SubElementSep	Separators coded in the map. Used only if "UseSep" property is setup to "true"
Filter	Can be setup to #13#10 to filter carriage return and line feed characters from the incoming data

Product supports both UN/EDIFACT and IATA EDIFACT for Airline Industry. In order to perform EDI translation you need to have separators defined at the root level of the message. In case if you do not know those separators, or you expect EDI files with various separators, use option "AutoDetect" and set it to "true". Separators are special characters that separate segments and elements in EDI document. You can find all three separators: sub element, element and segment separator, defined in the EDI X12 file at the location starting position 103 after ISA. If you use the wizard separators will be defined for you. In EDI X12 and EDIFACT mappings only elements should be mapped to output.

Typical EDI map might look something like this

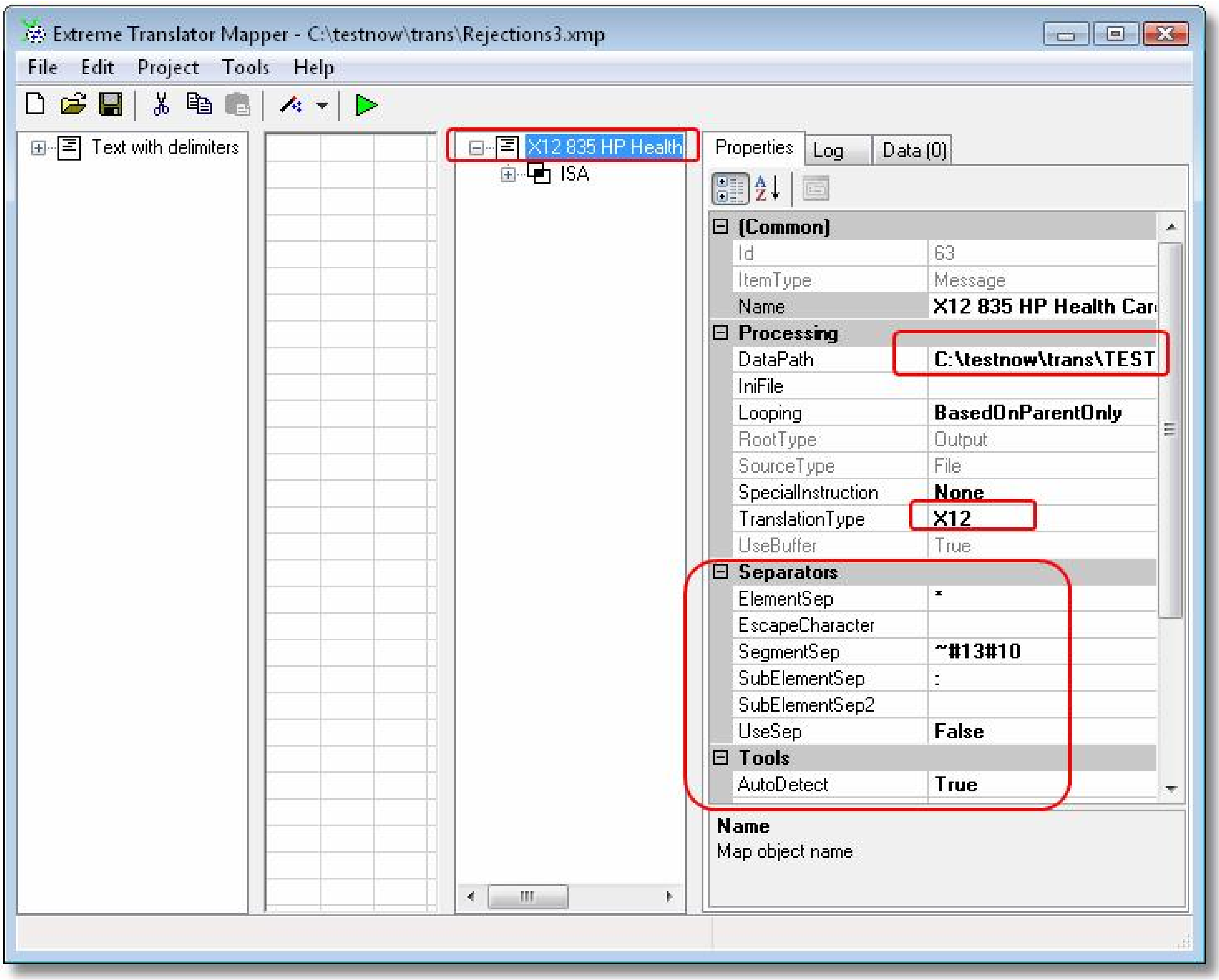


Format	
Mandatory	False
Sequence	
SpecialFur	None
Plugin	
ExternalFu	FlatFilter:Ex
Parameters	
Processing	
EndTag	|
LoopEndT	
SpecialInst	None
StartTag	
Processing Fixed Leng	
FixedLengt	False
Length	0
PadCharac	
PadType	Left
Script	
Script	
Special Processing	

SpecialInstruction
Special processing instruction for the translator

Flat file to EDI X12 mapping example.

There are certain properties on the message level that need to be setup for EDI files (X12, EDIFACT) to be processed. We will take simple fragment of EDIFACT file and do the mapping to flat file with output data elements separated by delimiters. First you need to setup DataPath property. We simply make it point to file in a local drive.


Extreme Translator Mapper - C:\testnow\trans\Rejections3.xmp
File Edit Project Tools Help
Text with delimiters
X12 835 HP Health
ISA
Properties
Log
Data (0)

(Common)	
Id	63
ItemType	Message
Name	X12 835 HP Health Car
Processing	
DataPath	C:\testnow\trans\TEST
IniFile	
Looping	BasedOnParentOnly
RootType	Output
SourceType	File
SpecialInstruction	None
TranslationType	X12
UseBuffer	True
Separators	
ElementSep	*
EscapeCharacter	
SegmentSep	~#13#10
SubElementSep	:
SubElementSep2	
UseSep	False
Tools	
AutoDetect	True

Name
Map object name

Properties that should be setup for EDI messages

Important properties relate to EDI separators are listed in Separators group. You should setup ElementSep, SegmentSep, SubElementSep if you setup UseSep to “true”. If you do not setup UseSep property to “true” you should set AutoDetect to “true” in that case translator will try to find separators in default locations: ISA header in EDI X12 or UNB header in EDIFACT. We also setup Filter property to filter carriage return and line feed characters (decimal codes 13 and 10).

This is a fragment of EDIFACT we want to translate:
UNB+UNOA:2+SGSIN+HOITD+021220:0953+SINFO019'
UNH+1+IFCSUM:D:01A:UN:UAS120'

Mapping is based on two segments on the input side called UNB and UNH. There are some composite data elements that have sub elements we need to extract. Examples would be Element 1 in UNB or Element 2 in UNH. In order to extract sub elements mappings should be edited using menu option “Mappings” on the popup menu when you click item on the left.

HIPAA Support

HIPAA is supported through the map templates. You can choose one and have map created in seconds. Those HIPAA templates are available

EDI X12 - 270 Eligibility, Coverage or Benefit Inquiry
EDI X12 - 271 Eligibility, Coverage or Benefit Inquiry

EDI X12 - 276 Health Care Claim Status Request
EDI X12 - 277 Health Care Claim Status Notification
EDI X12 - 278 Health Care Service Review
EDI X12 - 820 Payment Order Remittance Advice
EDI X12 - 834 Benefit Enrollment and Maintenance
EDI X12 - 835 Health Care Claim Payment Advice
EDI X12 - 837 Health Care Claim

Properties

You can see there are many properties describing each item in the map. As you click on the item all properties are displayed and you can change them. Some properties are common to all objects, such as *Name*, *Id*, *DataType*, *ItemType*.

There is the list of most used properties and they explanations:

Name	The name of map object, can be any name defined by the user. You should use names that are meaningful
DataType	It is the data type of the object. It can be "String", "Integer", etc. It is used for conversions and in special functions. You may leave it "undefined" unless you are going to use special functions
Id	Internal object ID used by translator
Format	See special chapter "Format Property"
Mandatory	This item must be present. This especially useful during export when you want certain elements produced in the file if other elements are produced. However you cannot make all the elements in the segment mandatory because it would create infinite loop during export
Sequence	Name of the sequence to be used to generate unique numbers. Useful at times when translator should produce EDI X12 control numbers. Other property called *IniFile* should be setup to get unique numbers.
IniFile	Path and file name of Initialization file. It will be used during translation to generate unique number and maybe used in other functions as well. Example: "C:\temp\editrans.ini".
SpecialFunction	Extra functions. You can use them to get total segment count, SE count used in EDI X12. Also get current input and output file names.
DataPath	Location of file or data for input or output. You can override this property by passing parameters in command line. File names may contain wildcards, such as * or ?. Example: C:\test\trans\inputxml* would pickup all the files having names start with "inputxml" in that directory. Wildcards cannot be used in directory part of path (see special chapter "DataPath Property").
Looping	BasedOnParentOnly – default looping model when each output item repeats in synchronization with its parent one level up. BasedOnAllParents –looping model when each output item repeats in synchronization with all of its parents up.
TranslationType	Translator performs certain optimizations on some file formats that it has special routines for performance and validation. "Txt" type is most generic and should be used only in cases when no other type matches.
UseBuffer	*Reserved for future use*. Means that translator will cache data in memory during processing. This is good option for small files, less than 1Mbt in size. This property will allow optimize translation for large files. Default is "true".
ElementSep, *SegmentSep*, *SubElementSep* *SubElementSep2* *EscapeCharacter*	Are separators used in some translations, such as EDI X12, EDIFACT. They separate data segments, such as ISA, GS, GE, UNA, UNB, etc. EscapeCharacter is used in EDIFACT for special cases when SegmentSep should be treated as simple data character and not a segment separator.

	SubElementSep2 is used in HL7.
UseSep	Indicator to use hard coded separators
AutoDetect	Automatically detect features of the file just before processing and use them during file processing. For example in cases if those files are EDI X12, EDIFACT files separators fill be detected automatically. This property overrides other properties such as *UseSep* property for example. However in some cases AutoDetect is impossible. Example: your EDI X12 files come without ISA header. In this case translator will not be able to find separators and translation will fail.
Filter	Can be used to filter garbage characters from the file or message. It is mostly used for removing CRLF (carriage return and line feed characters) from EDI X12 and EDIFACT messages, decimal codes #13#10. Some junk characters are being added by graphical file editors for display purposes and saved back into the file. In many cases they are not valid and do not contain any business data and serve no purpose during processing.
Substitute	Can be used to substitute characters in the input with other characters or strings of characters. Example of use: 1=TEST,2=TEST2 this will replace all instances of 1 with word TEST and all instances of 2 with word TEST2.
Remove	Can be used to remove strings or characters from the message. Regular Expressions can be used in this property.
TrimSpaces	Trim spaces from the end or start or both sides of each data element. Can be used for translation of fixed length flat files into other formats.
NullValues	If set to "Ignore" translator will skip incoming NULL values. If set to "TreatAsBlankValues" translator will treat NULL values as empty values. This setting is mostly used for mappings from database in cases when non existing field value should still be produced in the output as spaces or blank value.
ChangeCase	Change case on each data element. Can be used when translation should output data only in upper or lower case.
Encoding	Used to define of characters during translation. Use ASCII_7bit for English, European_8bit for European languages and Unicode for languages that are Unicode based.
SpecialInstruction (use of this property is also discussed in additional document related to looping issues and it is available as separate download on our website)	Tells translator to perform special operations during data translation FlatOutput – special instructions can be used on input elements. Tells translator to output looping element inline. See EDIX12-to-FlatFile example. It can also be used on output elements of flat file. See end of Text Translation chapter. AppendToFile – instruction to append output to file rather than overwriting and creating a new file (use on root item only). DeleteWhenDone – instruction to delete files after processing has finished (use on root item only). OutOfLoopData – instruction for translator do not pay attention to sequence of this element in the loop. It can be used on looping elements that are inside of the loop but data into them is coming from other loop (use on elements only). It should be used only as exception in cases when default looping does not work correctly. FlatOutputInline – it can be used on output element to make it line up correctly in the output file.
ConditionType, Condition	Two properties form simple conditional "IF" statement. (Currently implemented for TranslationType = X12, Edifact, XML). You can specify multiple conditions under one segment by simply entering them separated by commas.
ValidationType Validation	Type of validation to be performed. ValidationType is MustEqualTo then Validation should be the value you expect in the input ValidationType is SchemaFile then Validation should be path to XML Schema file for validation.

StartTag	It is a generic name for data block that indicates start of segment or element. If it is used for input it indicates boundaries where data starts. If it is used for output it indicates what will be placed to output before the data.
EndTag	It is an end marker for data block. Should be used for items that have no nested items attached. It is supplement to *LoopEndTag*.
LoopEndTag	It is an end marker for the block of looping data. Should be used for items that have nested items and those items have nested items too.
SQL	Free form SQL statement. It must be used in queries where SQLType set to Select. Can also be used in queries with SQLType InsertOrUpdate or Update. In this case SQL is added to WHERE clause for database Update. Example: if you want to update records only if they have field State=1, in this case your SQL should be “ AND STATE=1”.
SQLType	Select should be set for database query used on the input side. All other flag values are for database tables on the output side. Translator will use first PrimaryKey field in the database to perform Insert, InsertOrUpdate and Update operations. InsertOrUpdate is a combination of possible operations on the database where translator will select row from the database based on PrimaryKey, and if it exists translator will perform update otherwise insert will be performed.
MasterQuery	Name of other query in the map that will act as Master to form Master-Detail relationship and supply parameters to this query.
FieldType	Represents relational database enforced type. It is used to reorder and populate resulting records.
Size	Database field size.
FixedLength	Indicates to treat item as fixed length
Length	Actual length of item in characters
PadType	Dictates how fixed length item is padded with pad characters
PadCharacters	Character to be used for padding
ExternalFunction Parameters	Translator will call external routine stored in .NET assembly to get data. External routine must have certain function signature (see Plug-ins chapter in this document). Parameters will be passed into external function.
Script	Translator will call script that was defined via Scripts screen in Map Editor. Each script is compiled once before it is executed for the first time during map processing.

DataPath property

DataPath property is used on root items of the map for both input and output. It is generic name for data source that can be local file, file on the Internet accessible via http, ftp protocols or database connection string.
There are different rules what should be in this property depending on what is actual data source or destination. It is recommended to test processing using local files first and move them to ftp or http servers later.

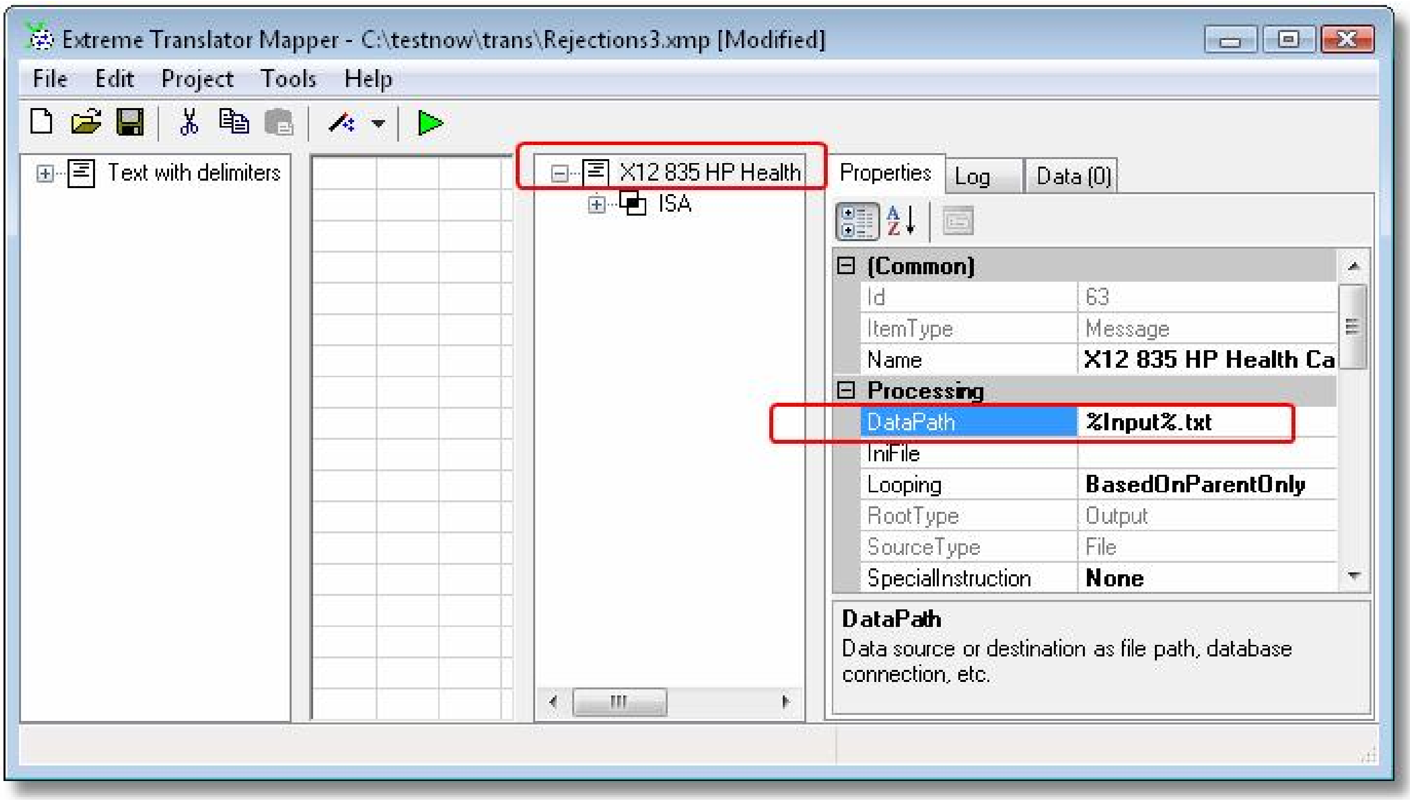
Local or network file(s)	DataPath property can be a path to the file including file name. File name also may contain wild cards like ‘*’ and ‘?’. Example: C:\test\trans\*.txt would pick up and translate all the text files in C:\test\trans directory. It can also be a shared directory on another computer in local network, like \\MYSERVER\testing\*.txt
Internet file	DataPath property can be an URL to the file on the web. It should start with string “http:”. Example: http://www.somewebsite.com/somedata.xml Translator would pick up XML file from the web site. Wild cards are not allowed. Translator uses HTTP GET to retrieve file and HTTP POST to submit files to the web server.
Database connection	DataPath property can be a connection string to the database. Example of connection string to ODBC data source:

	DSN=mydatasource;UID=myusername;PWD=mypassword
FTP file(s)	DataPath property can be a connection string to file on FTP server. Connection string can contain user name, password, FTP server address and remote server directory. On the input side DataPath may contain file pattern for the file to be picked up ftp://username:password@ftpserver.com/directory/filepattern Example: ftp://myuser:mypassword@someserver/test/.txt It would connect to "someserver" using user name "myuser" and password "mypassword", change to directory /test and pick up file that has ".txt" in the name. On the output side FTP file should be exact file name ftp://myuser:mypassword@someserver/test/myoutputfile.txt If login should be anonymous, user name and password might be omitted. Example will fetch all XML files from FTP server called "someserver" and remote directory "somedirectory" ftp://someserver/somedirectory/.xml SpecialInstruction = DeleteWhenDone can be used on input side in order to remove processed file from FTP server. SpecialInstruction = AppendToFile is not available with FTP on output side. Files with extension "txt" are transferred in text mode. Binary transfer mode is used for all other files.

Special macro can be used to place input file name into the output and have all the output file names formed based on input file names. Macros should be used on output DataPath property.

%Input%	Translator drops file extension from the input file, therefore any extension can be used in output DataPath property to form output file.
%InputFileName%	Translator drops file extension and directory from the input file, therefore any extension and directory can be used in output DataPath property to form output file.
%Count%	This macro is replaced by internal file count during processing.
%SystemDate%	This macros is replaced with current date in form CCYYMMDD
%SystemDateTime%	This macros is replaced with current date and time in form CCYYMMDDhhmm
%#propertyID%	This macro is replaced by actual processing data from input or output. Example: if DataPath is set to C:\test\output%#187645%.txt translator will replace %#187645% with the first value that item with property Id=187645 has during translation.

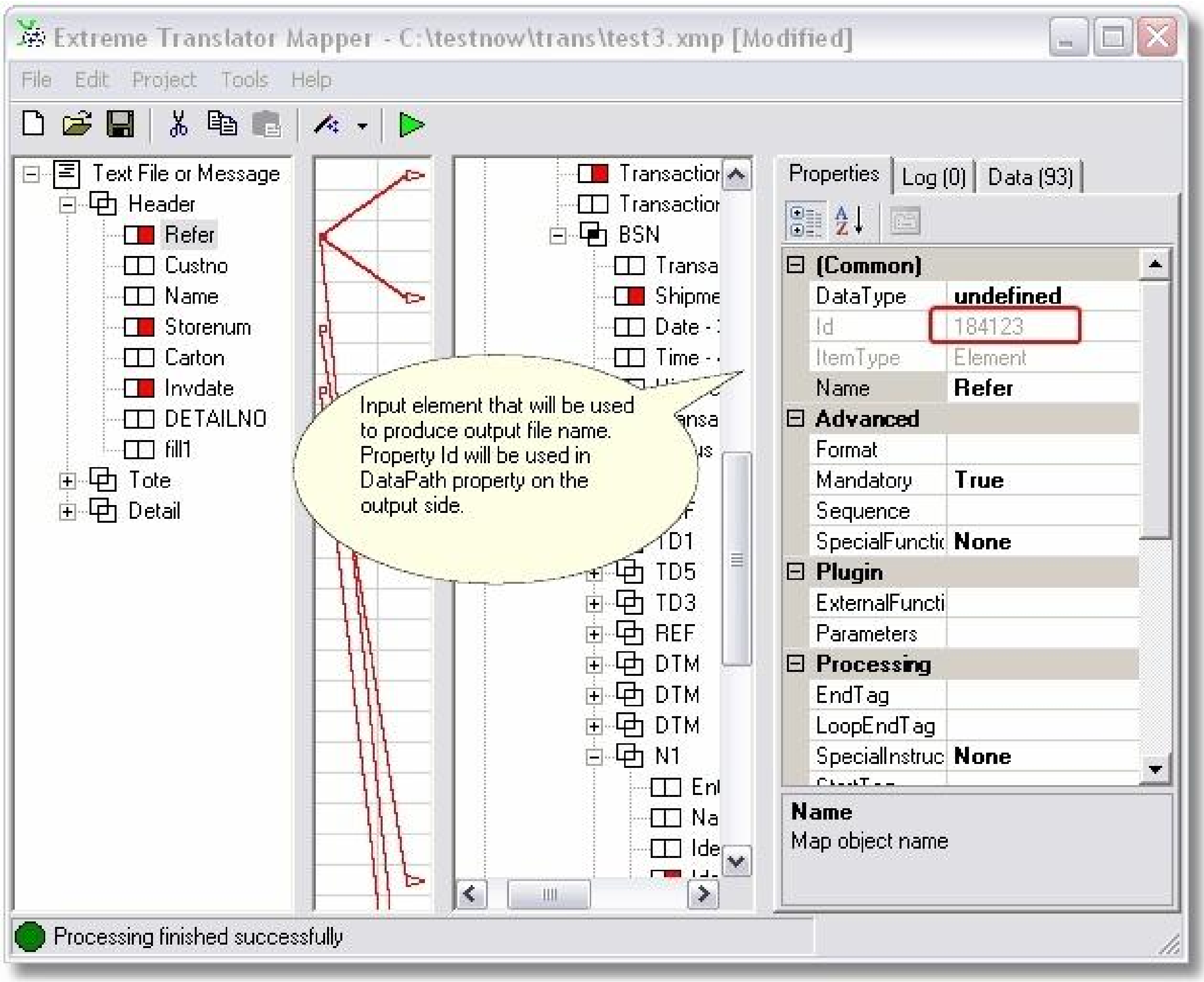
Important: Macros are case sensitive. They can be used only in one-to-one mappings when one type input message is mapped to one output message.



(Common)	
Id	63
ItemType	Message
Name	X12 835 HP Health Ca
Processing	
DataPath	%Input%.txt
IniFile	
Looping	BasedOnParentOnly
RootType	Output
SourceType	File
SpecialInstruction	None

DataPath
Data source or destination as file path, database connection, etc.

Example of macro %Input% usage.

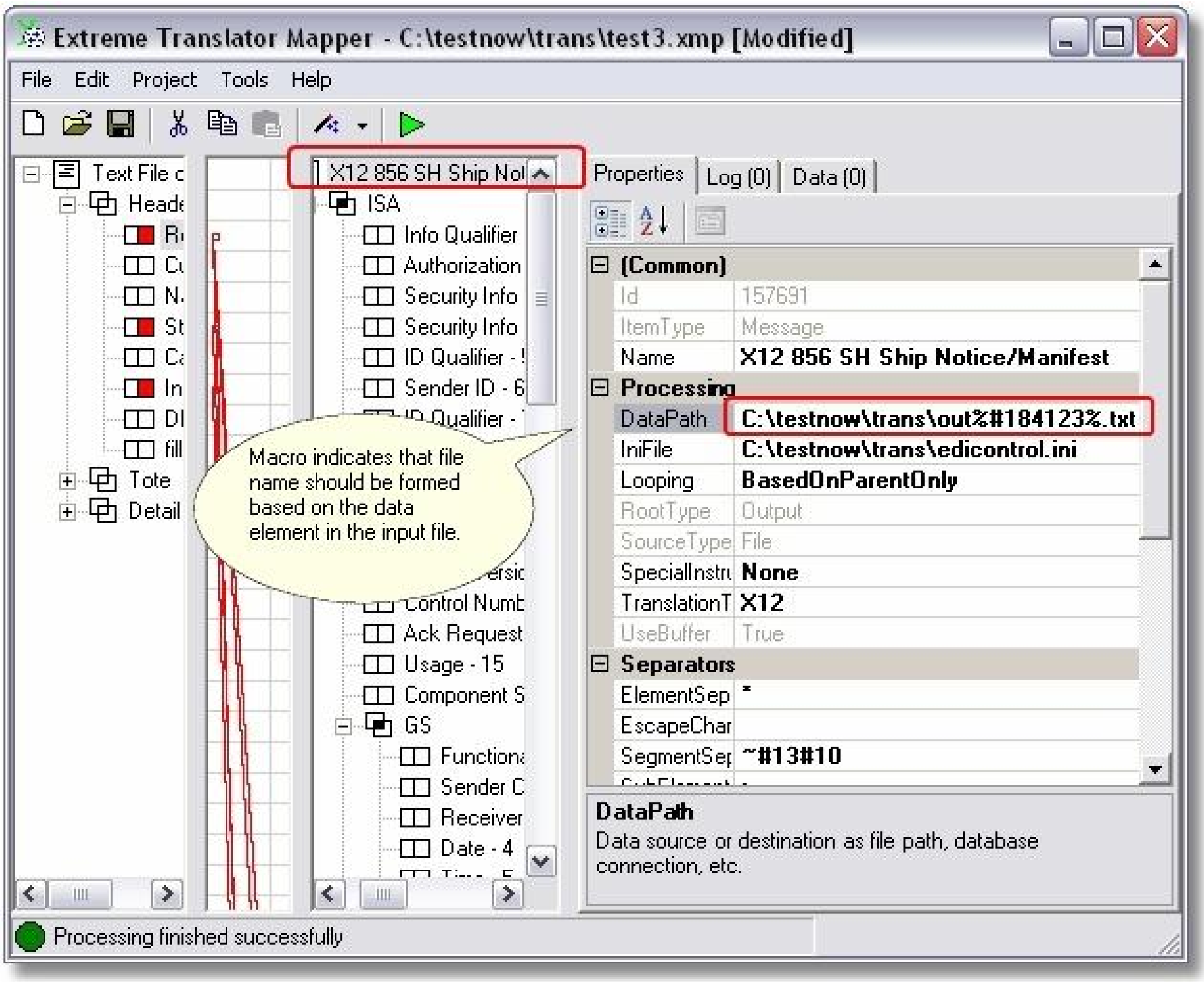


Input element that will be used to produce output file name. Property Id will be used in DataPath property on the output side.

(Common)	
DataType	**undefined**
Id	184123
ItemType	Element
Name	**Refer**
Advanced	
Format	
Mandatory	**True**
Sequence	
SpecialFunctic	**None**
Plugin	
ExternalFuncti	
Parameters	
Processing	
EndTag	
LoopEndTag	
SpecialInstruc	**None**

Name
Map object name

Example how %#propertyID% macro can be used to form DataPath based on the input or output data. See next screen shot for complete illustration.


Extreme Translator Mapper - C:\testnow\trans\test3.xmp [Modified]
File Edit Project Tools Help
X12 856 SH Ship Not
ISA
Info Qualifier
Authorization
Security Info
Security Info
ID Qualifier
Sender ID
Control Numb
Ack Request
Usage - 15
Component S
GS
Functiona
Sender C
Receiver
Date - 4
Properties
Log (0)
Data (0)
Macro indicates that file name should be formed based on the data element in the input file.
Processing finished successfully

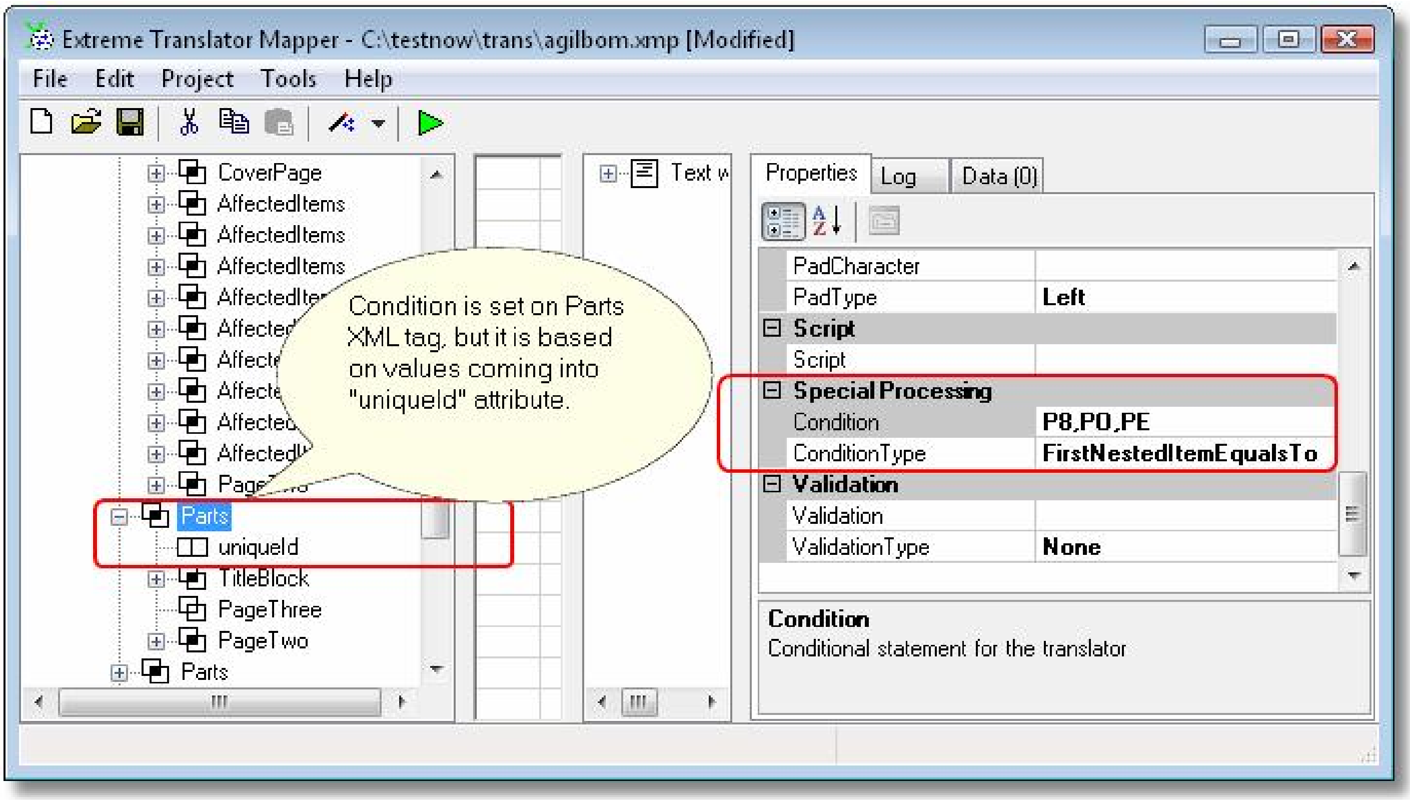
(Common)	
Id	157691
ItemType	Message
Name	X12 856 SH Ship Notice/Manifest
Processing	
DataPath	C:\testnow\trans\out%#184123%.txt
IniFile	C:\testnow\trans\edicontrol.ini
Looping	BasedOnParentOnly
RootType	Output
SourceType	File
SpecialInstru	None
TranslationT	X12
UseBuffer	True
Separators	
ElementSep	*
EscapeChar	
SegmentSep	~#13#10

DataPath
Data source or destination as file path, database connection, etc.

Example how %#propertyID% macro can be used.

Condition and ConditionType properties

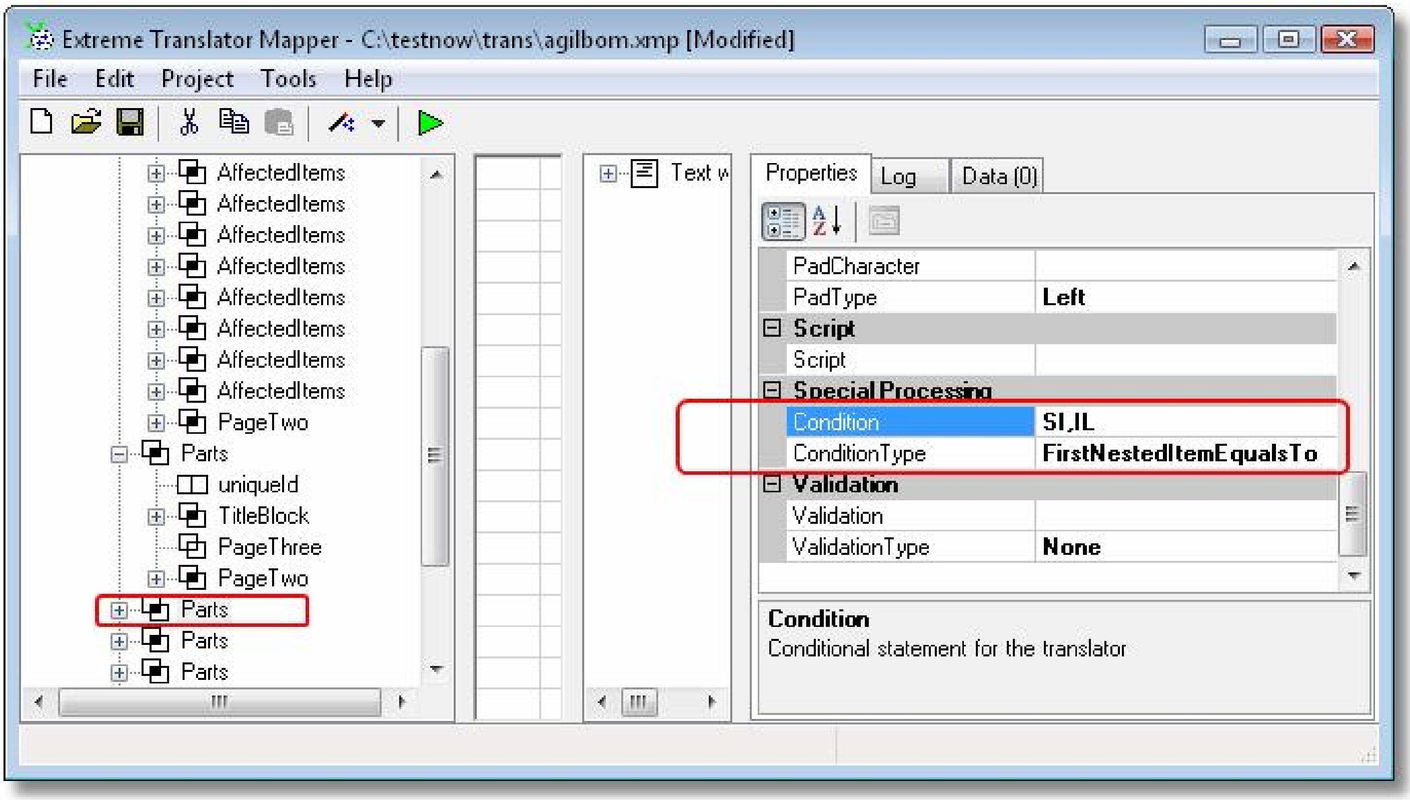
Properties Condition and ConditionType provide a way to filter or route incoming data to separate locations on the output file. In this example below only certain XML data with attribute “Qualifier” equal SI and IL should be placed into the output, and all XML tags that have “Qualifier” equal P8, PO and EQ should be filtered and will not make into the output.


Extreme Translator Mapper - C:\testnow\trans\agilbom.xmp [Modified]
File Edit Project Tools Help
CoverPage
AffectedItems
PageTwo
Parts
uniqueId
TitleBlock
PageThree
PageTwo
Parts
Text w
Properties
Log
Data (0)
Condition is set on Parts XML tag, but it is based on values coming into "uniqueId" attribute.

PadCharacter	
PadType	Left
Script	
Script	
Special Processing	
Condition	P8,PO,PE
ConditionType	FirstNestedItemEqualsTo
Validation	
Validation	
ValidationType	None

Condition
Conditional statement for the translator

In this example MiscRecord with P8, PO and EQ values is filtered and not placed into the output.



PadCharacter	
PadType	Left
Script	
Script	
Special Processing	
Condition	SI,IL
ConditionType	FirstNestedItemEqualsTo
Validation	
Validation	
ValidationType	None

Condition
Conditional statement for the translator

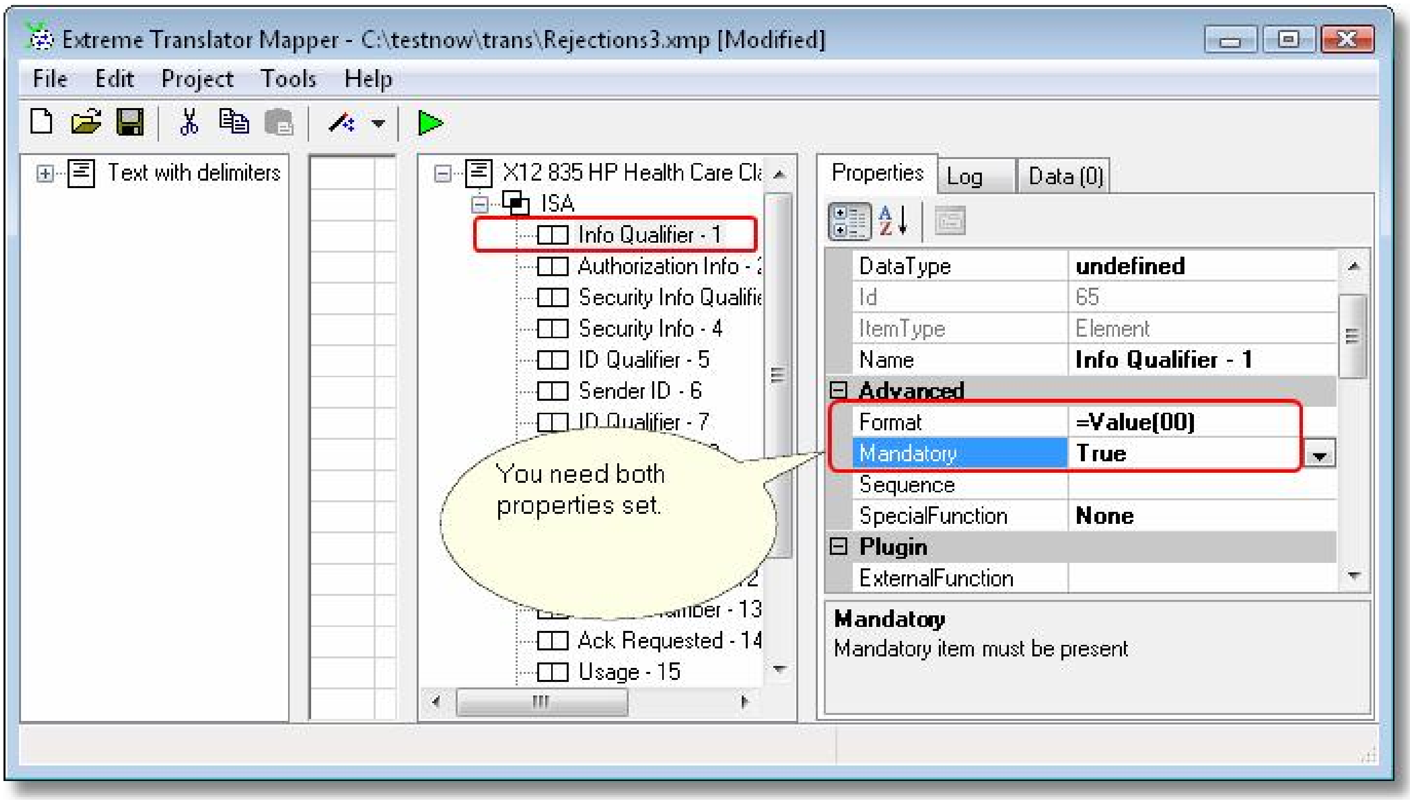
Only MiscRecord's with SI and IL are placed in the output.

Format property

It is advanced property that allows special formatting to be applied to data during translation. It is powerful feature. There are number of types of expressions you can use in this property:

Function	Description
=Value()	Place constant value into the element or field.
=ValueIfNull()	Place constant value if data for that element or field is NULL (no data).
=ValueIfNotNull()	Place constant value if data for that element or field is not NULL (has data).
=Form()	Reformat data.
=Match()	Match using Regular Expressions.
=Replace(;)	Replace using Regular Expressions.
=Substitute()	Substitute certain values based on the list provided.
=Evaluate()	Perform simple arithmetical operations on data, append or add additional character data.

Value is used to place default value in the map item during processing.
Example of use: =Value(defaultvalue).
ValueIfNull is used to place default value in the map only if incoming data is NULL or empty string.
ValueIfNotNull is opposite of ValueIfNull.



DataType	undefined
Id	65
ItemType	Element
Name	Info Qualifier - 1
Advanced	
Format	=Value(00)
Mandatory	True
Sequence	
SpecialFunction	None
Plugin	
ExternalFunction	

Mandatory
Mandatory item must be present

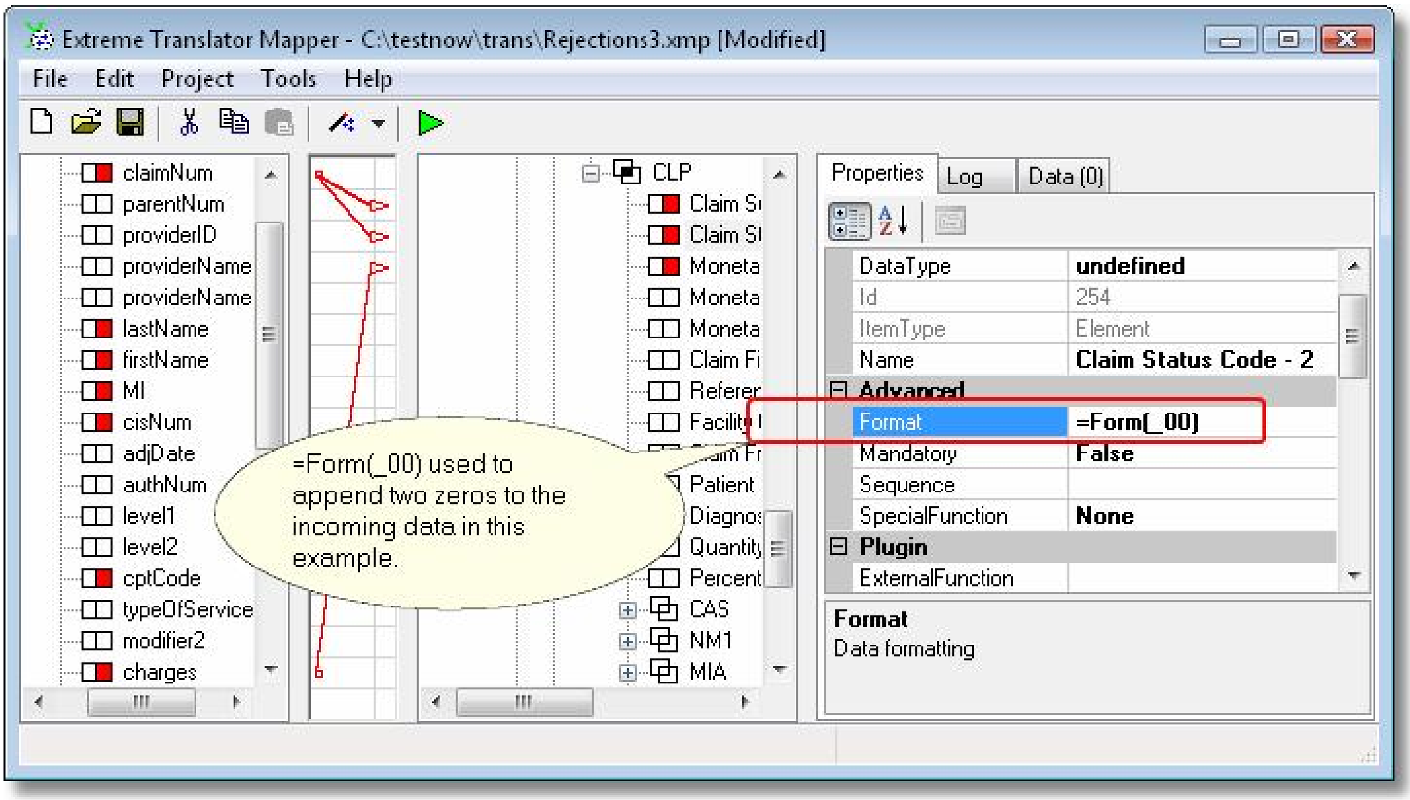
Default value placement in output.

Form is used to form output data based on input.
You may use special escape characters to manipulate data. Escape characters are: “@” and “_”.
Usage:
@Position number - may look like this "00@21" would print "00" + character at position 21 in the data. Position number count starts from 1, not 0.
_ - will place all data in this location, so "00_" would print "00" + all the data

Example 1:
You have data in format YYYYMMDD coming from EDI X12 file. You want to convert data into format MM/DD/YYYY. In order to do that you can setup *Format* property like this:
=Form(@5@6/@7@8/@1@2@3@4)

This basically says: take character at position 4 from input and place in position 0, then take character at position 5 and place at position 1, place "/", take character 6 and place at position 4, etc. If you need to add some constant data to the output of some element, simply use "_".

Example 2:
You want to add four zeros in at the end of data. Your *Format* would look:
=Form(_0000)



DataType	undefined
Id	254
ItemType	Element
Name	Claim Status Code - 2
Advanced	
Format	=Form(_00)
Mandatory	False
Sequence	
SpecialFunction	None
Plugin	
ExternalFunction	

Format
Data formatting

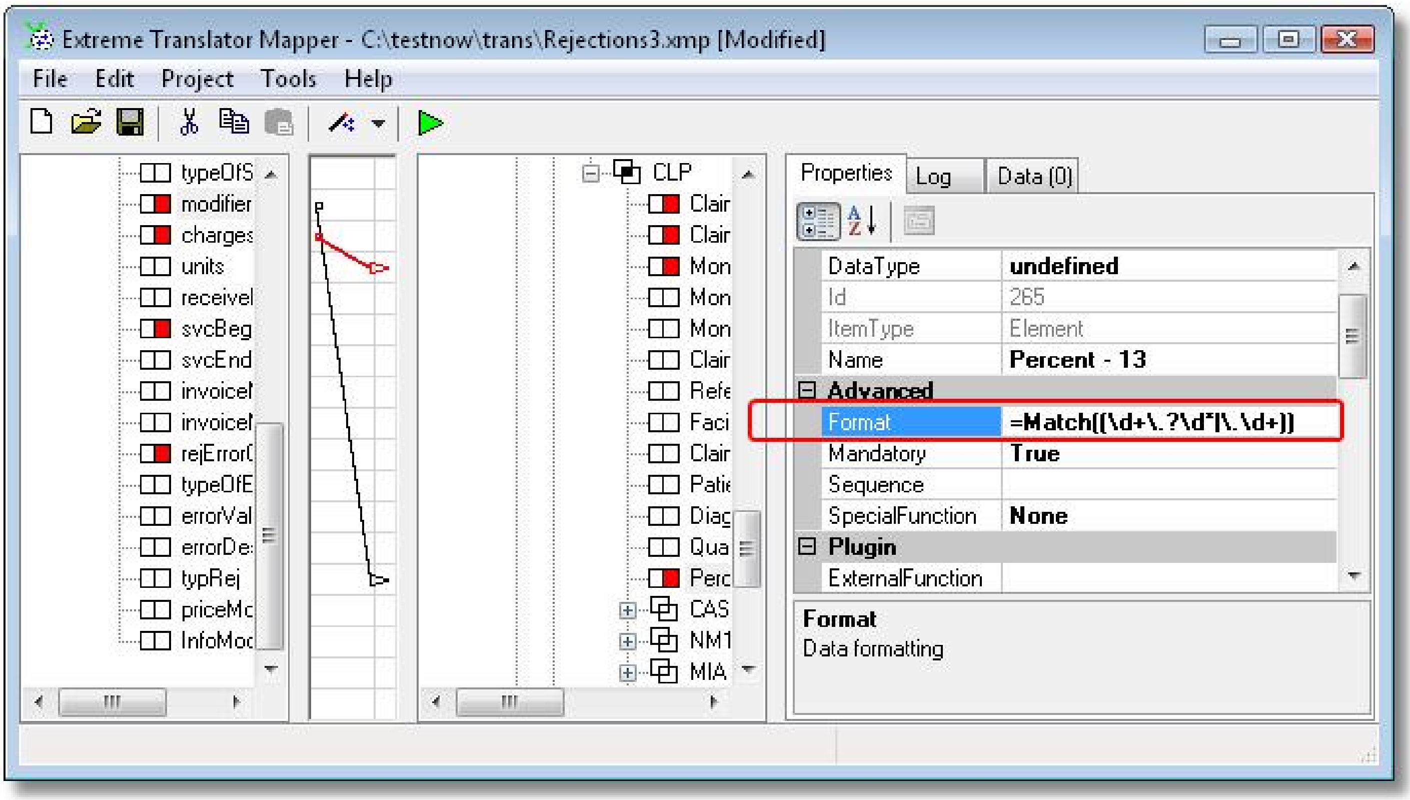
Example of "=Form()" usage.

Match and Replace is based on regular expressions processing used in Perl, awk and many other utilities and languages. There are books written on how to write powerful regular expressions. Translator supports a subset of all available regular expressions.
Those are some of examples on how to use "=Match()" and "=Replace(;)".

If input is "abracadabra" then "=Match((a|b|r)+)" would give us "abra" which is the amount of string that has been successfully matched.
If input is "abracadabra" then "=Replace(zzzz;abra)" would give us "zzzzcadzzzz", in which all occurrences of the matching pattern are replaced by the replacement string "zzzz".
Note: semicolon is used to separate replacement string "zzzz" and actual regular expression "abra" in previous "=Replace".
If semicolon has to be in your input data it can be escaped with backslash character. Example: =REPLACE(&APOS;;') would replace &APOS with ;' but your desired result is to replace &APOS; with ' (quote), so your Format property should be =REPLACE(&APOS\;;').

Examples of some regular expressions:
"=Replace(;^\s+)" remove leading whitespace
"=Replace(;\s+$)" remove trailing whitespace
"=Replace(;\s*\r?\n\s*)" joining lines in multilane strings (or removing carriage return and line feed)
"=Match((\d+\.?\d*|\.\d+))" extract all numbers from string

As you can see regular expressions can be very powerful however somewhat cryptic to read and use. You may consider searching for more information on them in Internet sources.



DataType	**undefined**
Id	265
ItemType	Element
Name	**Percent - 13**
Advanced	
Format	**=Match((\d+\.?\d*|\.\d+))**
Mandatory	**True**
Sequence	
SpecialFunction	**None**
Plugin	
ExternalFunction	

Format
Data formatting

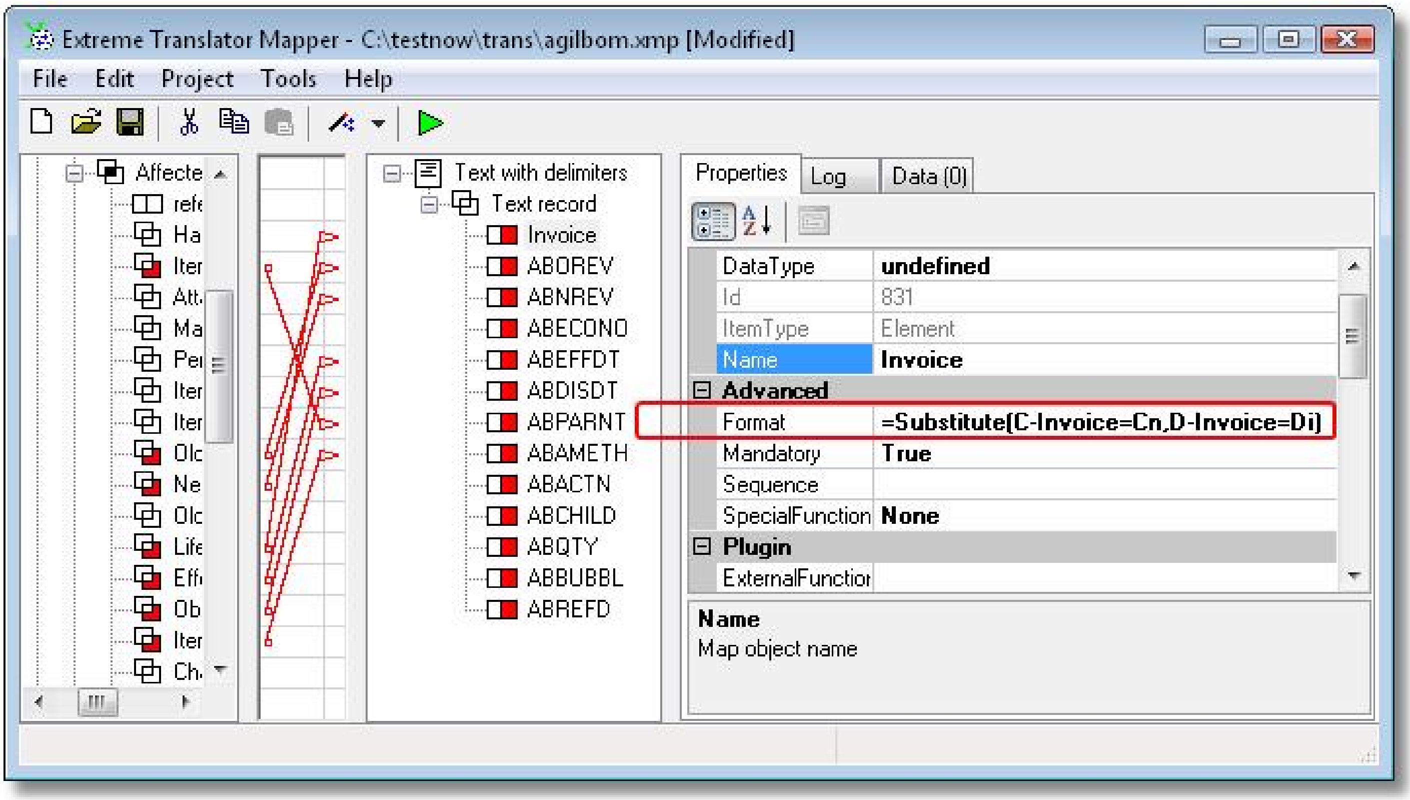
Example on using "=Match()" regular expression.

The regular expression language includes two basic character types: literal (normal) test characters and metacharacters. Regular expression metacharacters are an evolved extension of the ? and * metacharacters used with the MS-DOS file system to represent any single character or group of characters.

This is a short overview of common regular expressions metacharacters:

Expression	*Meaning*
.	Matches any character except \n
[characters]	Matches a single character in the list
[^characters]	Matches a single character not in the list
[charX-charY]	Matches a single character in the specified range
\w	Matches a word character; same as [a-zA-Z_0-9]
\W	Matches a nonword character
\s	Matches a whitespace character; same as [\n\r\t\f]
\S	Matches a nonwhitespace character
\d	Matches a decimal digit, same as [0-9]
\D	Matches a nondigit character
^	Beginning of the line
$	End of the line
\b	On a word boundary
\B	Not on a word boundary
*	Zero or more matches
+	One or more matches
?	Zero or one matches
{n}	Exactly n matches
{n,}	At least n matches
{n,m}	At least n but no more than m matches
()	Capture matched substring
(?<name>)	Capture matched substring into group name
|	Logical OR

Substitute can be used to replace specific input values with predefined output values. Property format is =Substitute(oldvalue1=newvalue1,oldvalue2=newvalue2,*=newvalue3)
Special character "*" star means any other value will be newvalue3.



DataType	undefined
Id	831
ItemType	Element
Name	Invoice
Advanced	
Format	=Substitute(C-Invoice=Cn,D-Invoice=Di)
Mandatory	True
Sequence	
SpecialFunction	None
Plugin	
ExternalFunctior	

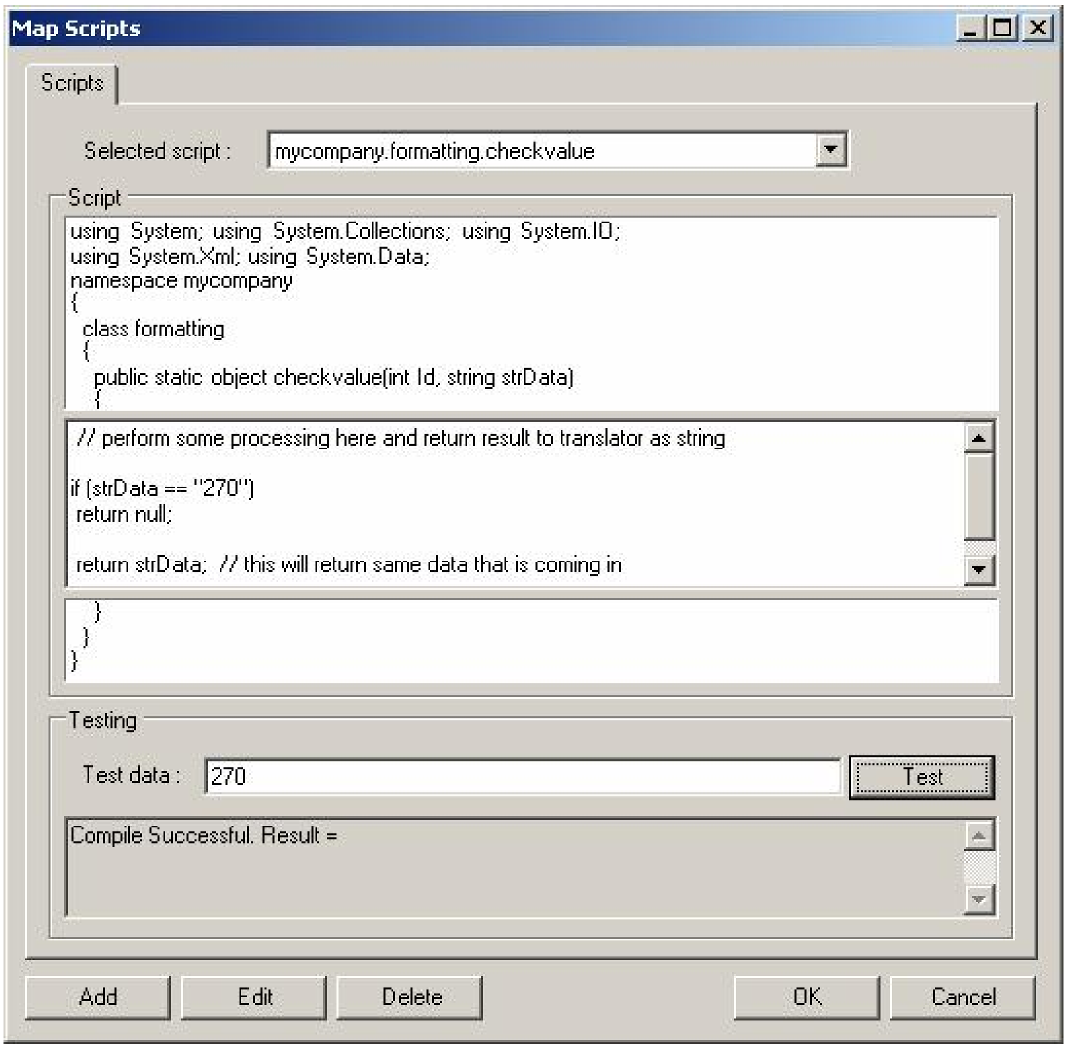
Name
Map object name

In this example if input is "C-Invoice" it will be replace to "Cn", but if it is "D-Invoice" it will be "Di" on output.

Evaluate can be used to perform simple arithmetical operations or help to concatenate character data. Special character @ can be used to indicate incoming data.
Example 1:
=Evaluate(10+@) will add 10 to incoming data. So if data that is coming to element or field is for example equal to 85 then result after evaluation will be 95.
Example 2:
=Evaluate(4 + @ + 'AAA') will add 4 to incoming data and append AAA to the end result. So if incoming data is 34 then end result would be 38AAA.
Example 3:
=Evaluate('ABC' + @ + 'DEF') will append ABC characters to the front of incoming data and append DEF characters to the end.
Example 4:
=Evaluate(@ / 10) will divide incoming number by 10.

Script property

If Format property cannot format data the way you want, you can use full power of C#.NET language to do extra processing using Script property. Add scripts via "Scripts" menu in Map Editor, then click on specific item that has Script property and assign script from the list.



Selected script : mycompany.formatting.checkvalue

Script

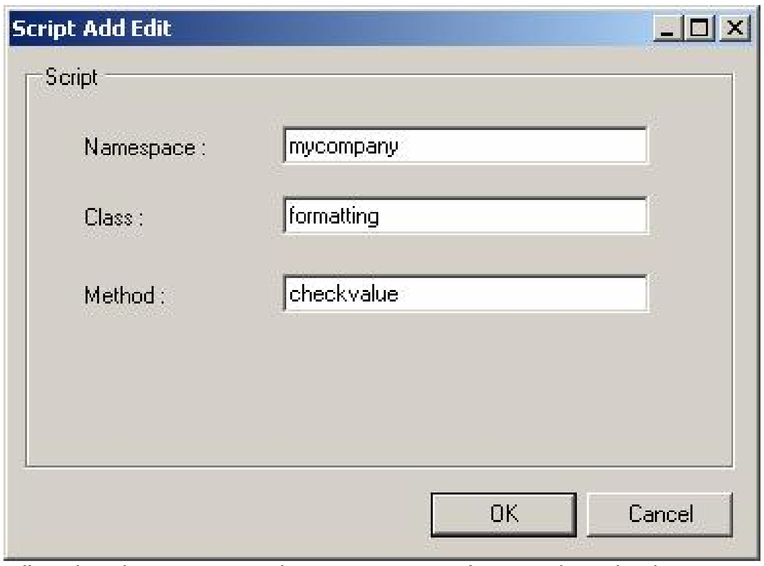
```
using  System;  using  System.Collections;  using  System.IO;
using System.Xml; using System.Data;
namespace mycompany
{
  class formatting
  {
    public static object checkvalue(int Id, string strData)
    {
 // perform some processing here and return result to translator as string

if (strData == "270")
 return null;

 return strData;  // this will return same data that is coming in
    }
  }
}
```

Testing

Test data : 270

```
Compile Successful. Result =
```

You can use Map Scripts dialog to add and test new scripts. If you click Test button script will be compiled and executed with data from “Test data” edit box assigned to strData variable.



Namespace : mycompany

Class : formatting

Method : checkvalue

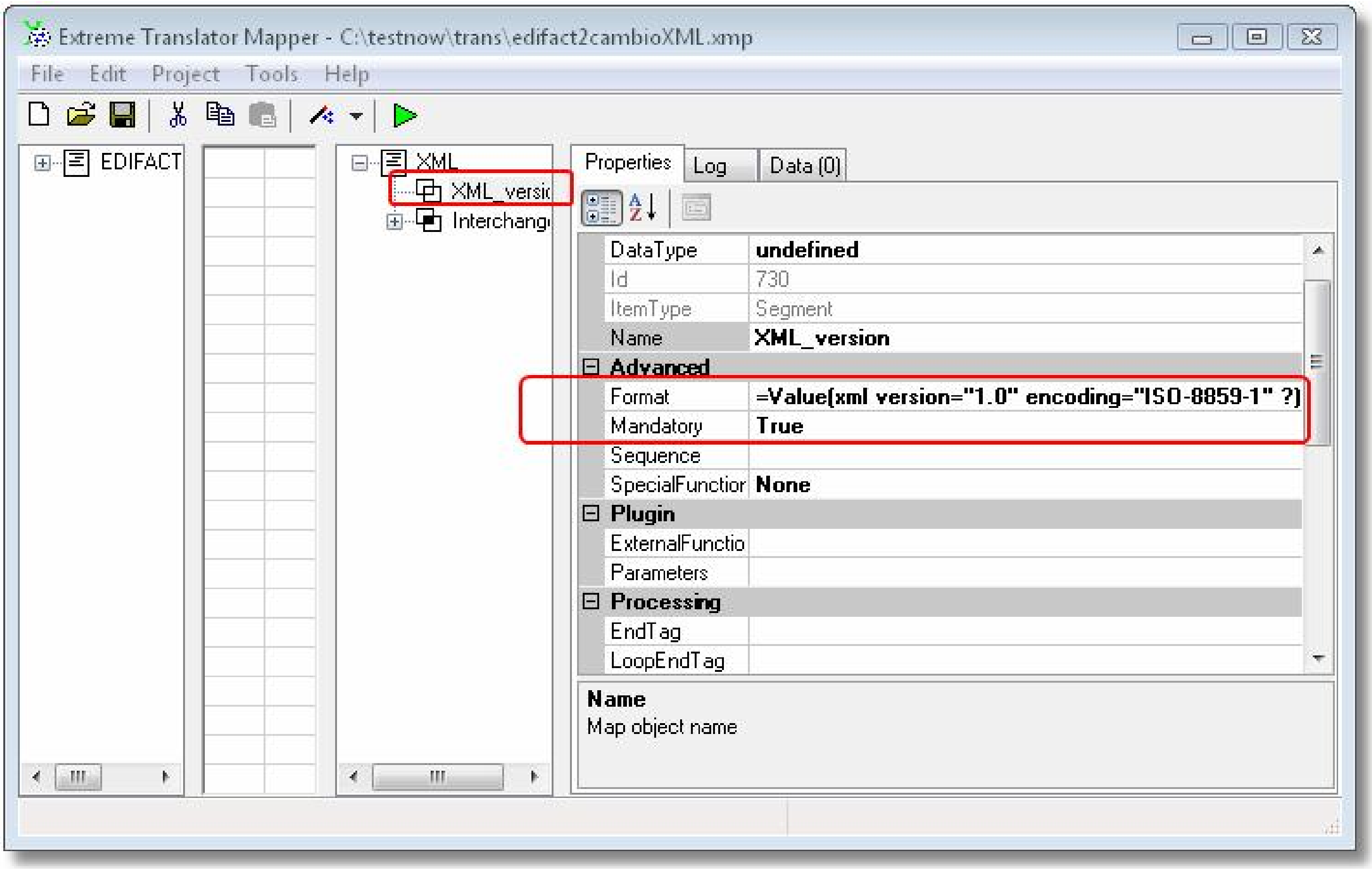
All scripts have to contain namespace, class and method names. For simplicity you can use your company name as namespace. Class and method names cannot match.

XML Translation

List of most important properties for XML maps

StartTag	XML tag or XML attribute name. Do not use "<" and ">" brackets in this property
EndTag, LoopEndTag	Not used
ValidationType, Validation	If ValidationType is set to SchemaFile then Validation should point to actual XML Schema file

Translator uses DOM parser included in .NET Framework.
Warning: XML file should have proper valid XML format including first XML item, example:
<?xml version="1.0"?>.
If this item is missing By-Example Wizard would not be able to setup map items based on XML file provided.


Extreme Translator Mapper - C:\testnow\trans\edifact2cambioXML.xmp
File Edit Project Tools Help
EDIFACT
XML
XML_versic
Interchang
Properties
Log
Data (0)
Name
Map object name

DataType	undefined
Id	730
ItemType	Segment
Name	XML_version
Advanced	
Format	=Value(xml version="1.0" encoding="ISO-8859-1" ?)
Mandatory	True
Sequence	
SpecialFunctior	None
Plugin	
ExternalFunctio	
Parameters	
Processing	
EndTag	
LoopEndTag	

Screen shot on how to setup XML attribute for the output.

Segments as well as elements can be mapped to output. Segment might look like:

<segment_name>some data</segment_name>

Elements are inside of segments and look like:

<segment_name elem1="some element data">
some segment data
</segment_name>

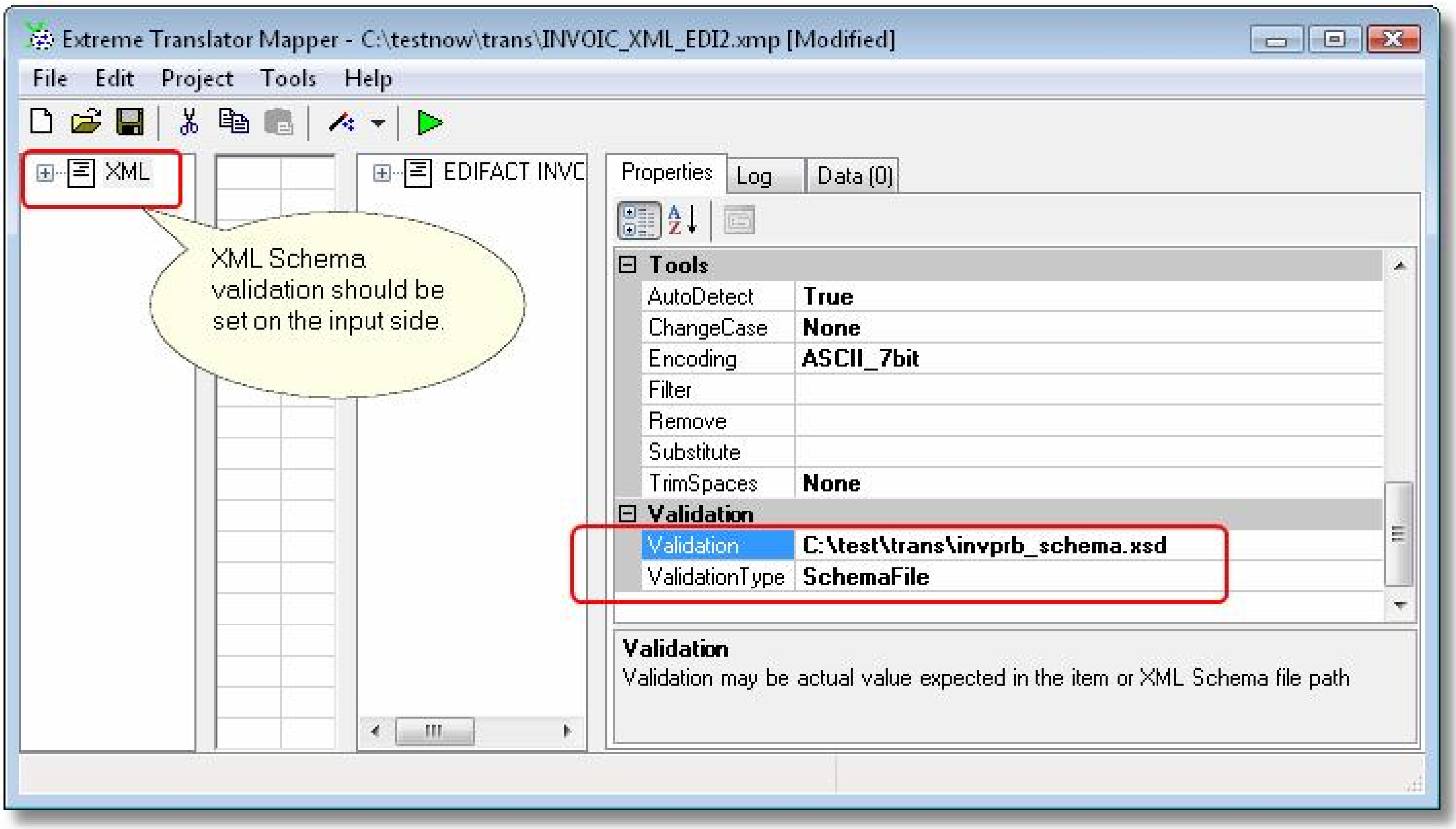
When you create mappings to produce XML, certain characters will be escaped in order to produce valid XML output. There is a table of escaped characters:

Input character	*Output string*
‘ (single quote)	&apos;
“ (double quote)	&quot;
> (greater than sign)	&gt;
< (less than sign)	&lt;
& (ampersand)	&amp;

XML Schema Validation

Validation against XML Schema can be performed using XML Schema file. Validation and ValidationType properties have to be setup to perform the validation.

By-Example Wizard can be used to create the map for you. After you run the wizard you have to click on the root item and fill *DataPath* and *Validation*, *ValidationType* properties.


Extreme Translator Mapper - C:\testnow\trans\INVOIC_XML_EDI2.xmp [Modified]
File Edit Project Tools Help
XML
EDIFACT INVC
Properties Log Data (0)
XML Schema validation should be set on the input side.

Tools
AutoDetect True
ChangeCase None
Encoding ASCII_7bit
Filter
Remove
Substitute
TrimSpaces None
Validation
Validation C:\test\trans\invprb_schema.xsd
ValidationType SchemaFile

Validation
Validation may be actual value expected in the item or XML Schema file path

Example of setting XML validation on the incoming XML file using XML Schema.

Flat Text File Translation

List of most important properties for delimited variable length text file processing maps

StartTag	Block of text or single character that marks the start of the item. You can think of it as an opening bracket in the text data. Most of the time it is delimiter, such as comma, new line character, space character, etc.
EndTag, LoopEndTag	Block of text that marks the end of the item. You can think of it as closing bracket. Translator will extract data between StartTag and EndTag/LoopEndTag.
SpecialInstruction	It can be setup to FlatOutput or FlatOutputInline on output

	element to make data in the output line up correctly. Please read to the end of this chapter for illustration.

List of most important properties for fixed length text file processing maps

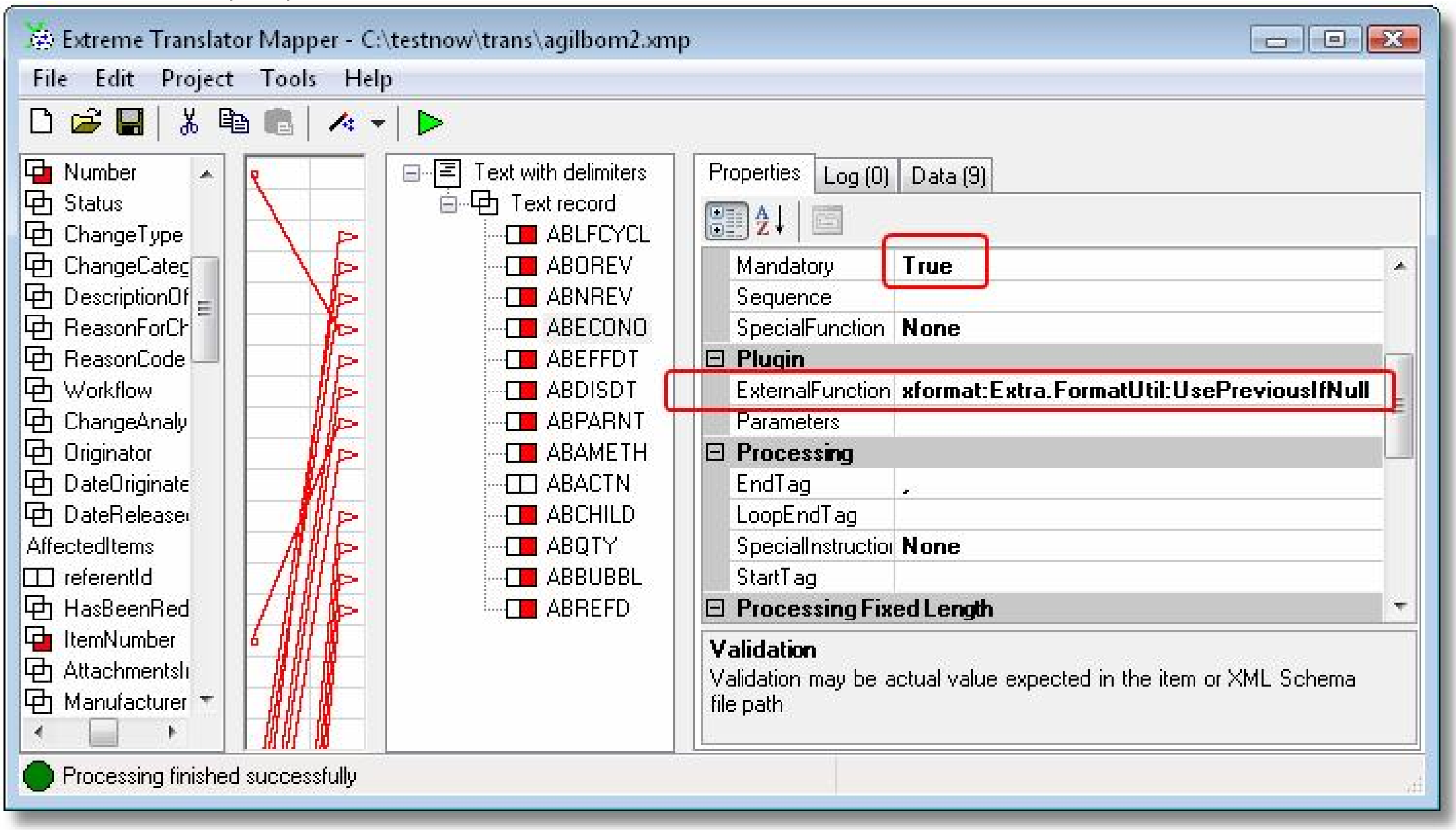
FixedLength, Length	If FixedLength is true the Length of actual data item to extract
PadCharacter, PadType	Actual character to be used if data should be padded on the Left or Right
Format	Can be used to produce desired output in a special format

Text translation is raw data conversion. It is the same model used in our first product called Import/Export Studio. This is less effective method than XML or EDI translations mainly because translator steps though every character in the file and does numerous iterations in the map to determine what this block of data is. Text files can be of any structure: flat files, fixed length files, files exported from Excel or MS Access, even dBase format DBF files. Any file that has a structure can be converted. Even EDI and XML files could be translated however translation would be raw file conversion not taking into account EDI and XML file specifics.

There are two most important properties you have to setup in the map in order to process delimited text files. Those properties are called “StartTag” and “EndTag/LoopEndTag”. You can think about these two properties as open and closing brackets. Just imagine brackets in your text file you want to process and based on this create segments and elements.
”StartTag” is the block of text translator will compare incoming data against. If data matches then translator will try to find “EndTag/LoopEndTag” and if it can find the end, it will step one level down into the map and try to locate nested items.
If you are trying to process fixed length files then you should use FixedLength, Length, PadCharacter and PadType properties.

It is possible to intermix text file processing and have both fixed length and delimited text processing techniques in one map but it is very rare.

Mapping XML, EDI X12 and EDIFACT formats to flat files can be a challenge just because they are nested and/or looping. Most of flat files contain some header and detail elements that are separated by commas or presented in fixed length. Header elements have to repeat per each detail line.

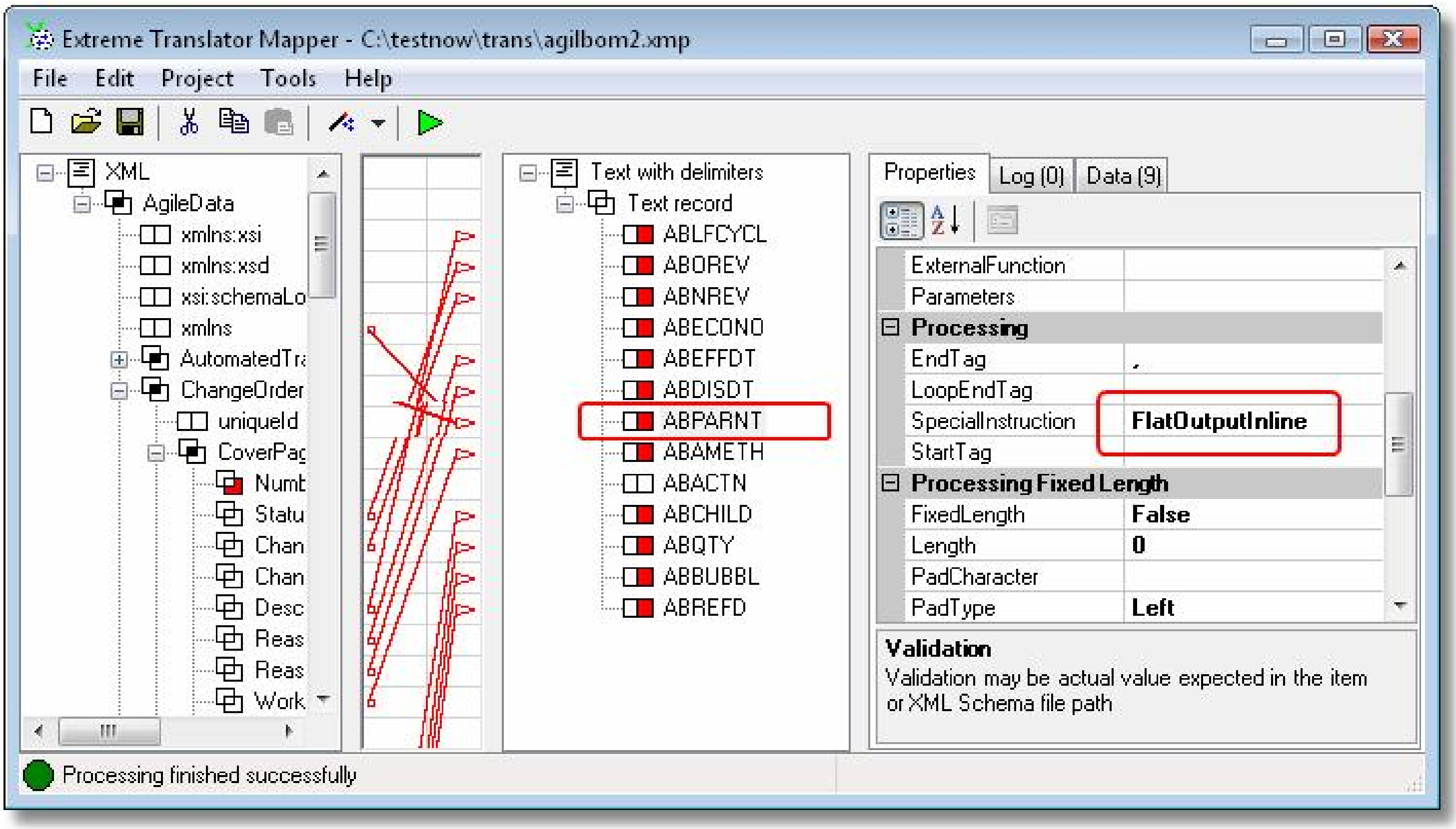


Mandatory	True
Sequence	
SpecialFunction	None
Plugin	
ExternalFunction	xformat:Extra.FormatUtil:UsePreviousIfNull
Parameters	
Processing	
EndTag	,
LoopEndTag	
SpecialInstructio	None
StartTag	
Processing Fixed Length	

Validation
Validation may be actual value expected in the item or XML Schema file path

Processing finished successfully

There is example of XML mapping to flat text file with comma delimiters. In this example header data has to repeat per each line of detail, so UsePreviousIfNull plug-in function is used to fill in empty spots.



ExternalFunction	
Parameters	
Processing	
EndTag	,
LoopEndTag	
SpecialInstruction	**FlatOutputInline**
StartTag	
Processing Fixed Length	
FixedLength	**False**
Length	**0**
PadCharacter	
PadType	**Left**

Validation
Validation may be actual value expected in the item or XML Schema file path

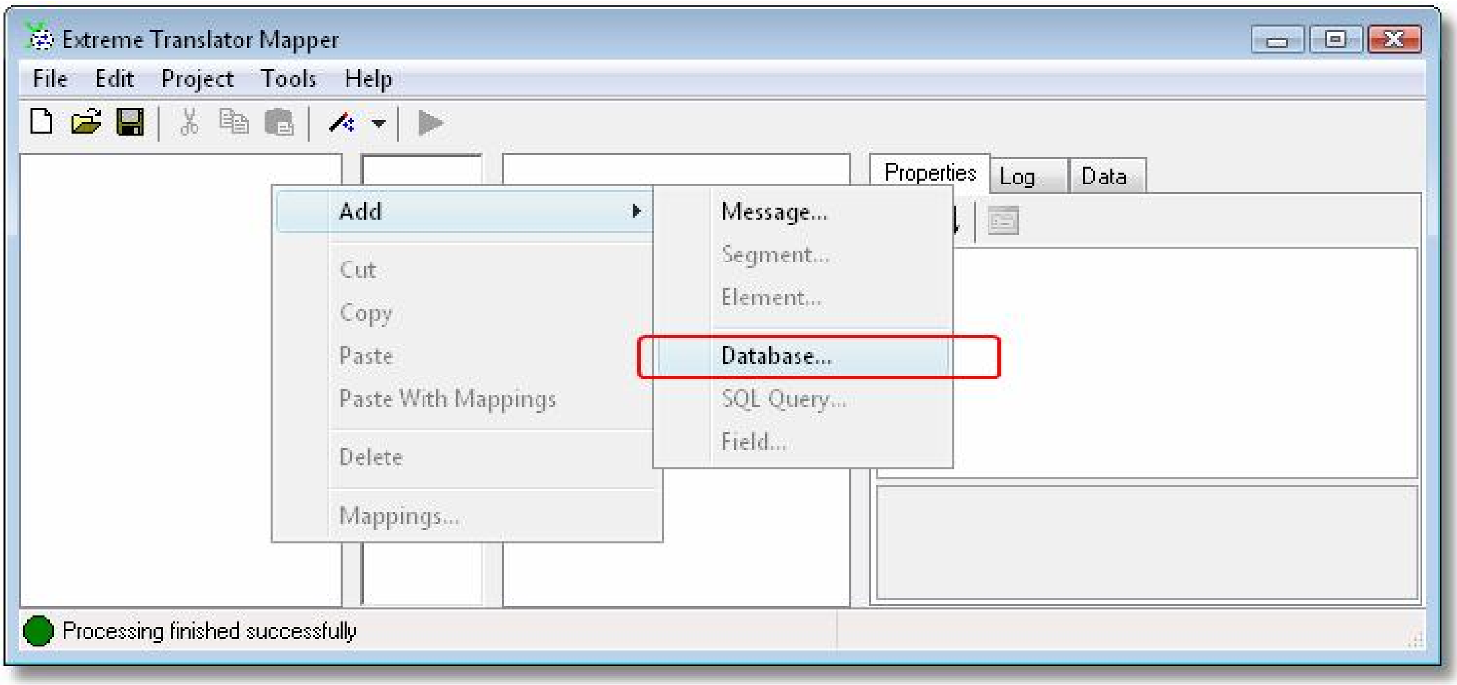
Processing finished successfully

Also to have all the headers line up with details SpecialInstruction property has to be setup to FlatOutputInline on first detail item in the flat file.

Database Mappings

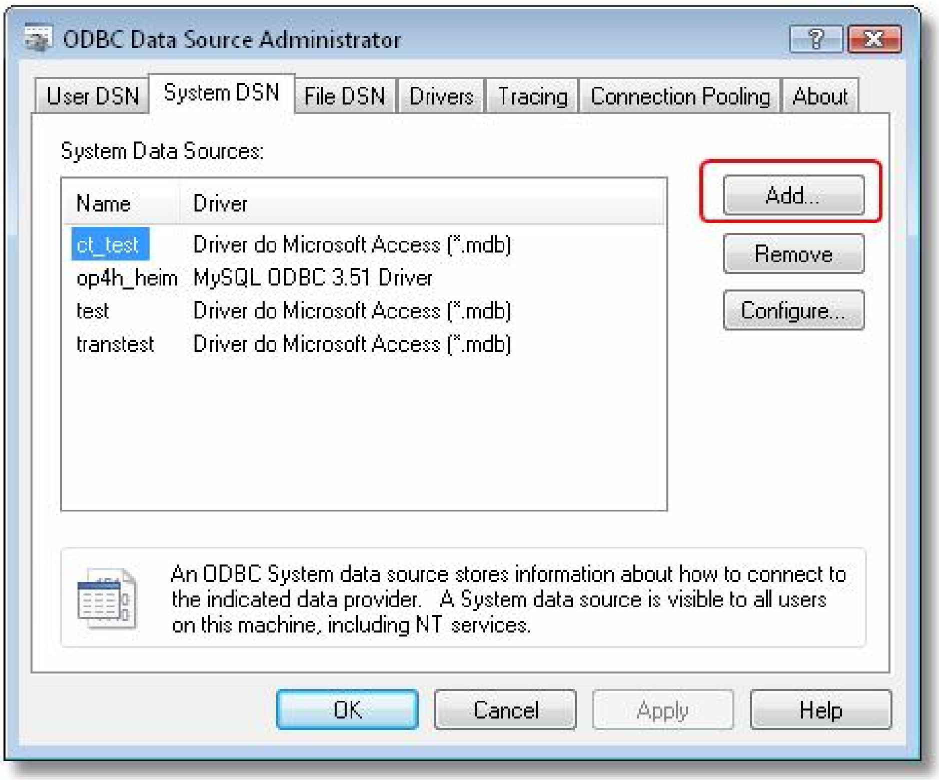
Mapping

Translator is capable to import or export data from database. ODBC connection is established to read data using SQL statements or write data into database tables. You have to define database objects in the map in order to process them in translator. Root item database object can be defined using popup menu “Add” and “Database”.



Adding database to the map.

Add Database dialog contains connection string information: ODBC data source name, user name and password. ODBC data source should be defined in Control Panel under Data Sources (ODBC).

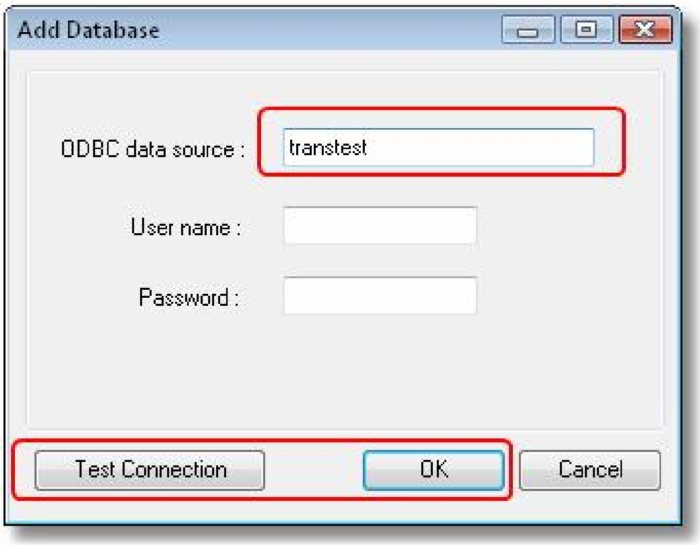


Name	Driver
ct_test	Driver do Microsoft Access (*.mdb)
op4h_heim	MySQL ODBC 3.51 Driver
test	Driver do Microsoft Access (*.mdb)
transtest	Driver do Microsoft Access (*.mdb)

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

Windows ODBC data source manager.

Sometimes databases do not require user name and password in that case it should be left blank. You can use “Test Connection” button to make sure connection works. After you press “OK”, connection information is used to populate “DataPath” property. This property can be modified later if data source name, user name or password changes.



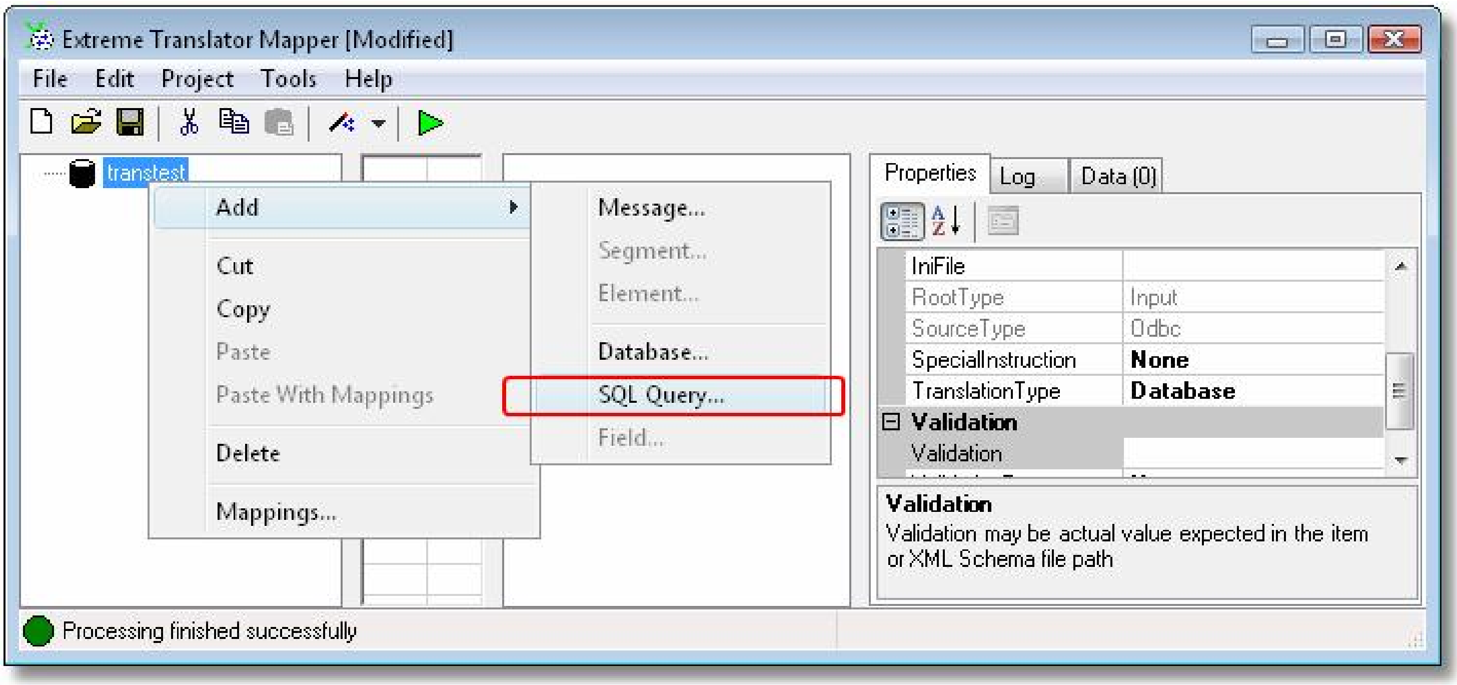
ODBC data source : transtest

User name :

Password :

Database connection settings.

When database is setup, you can add SQL queries to it. Queries you can setup on the left side (input) are different than ones you would setup on the right (output). Input queries are SQL SELECT based queries for data retrieval. Output queries are for SQL INSERT statements. Major difference is that you can edit SQL statement of SELECT query and modify it, when INSERT statement is dynamically constructed based on the fields listed under the query.



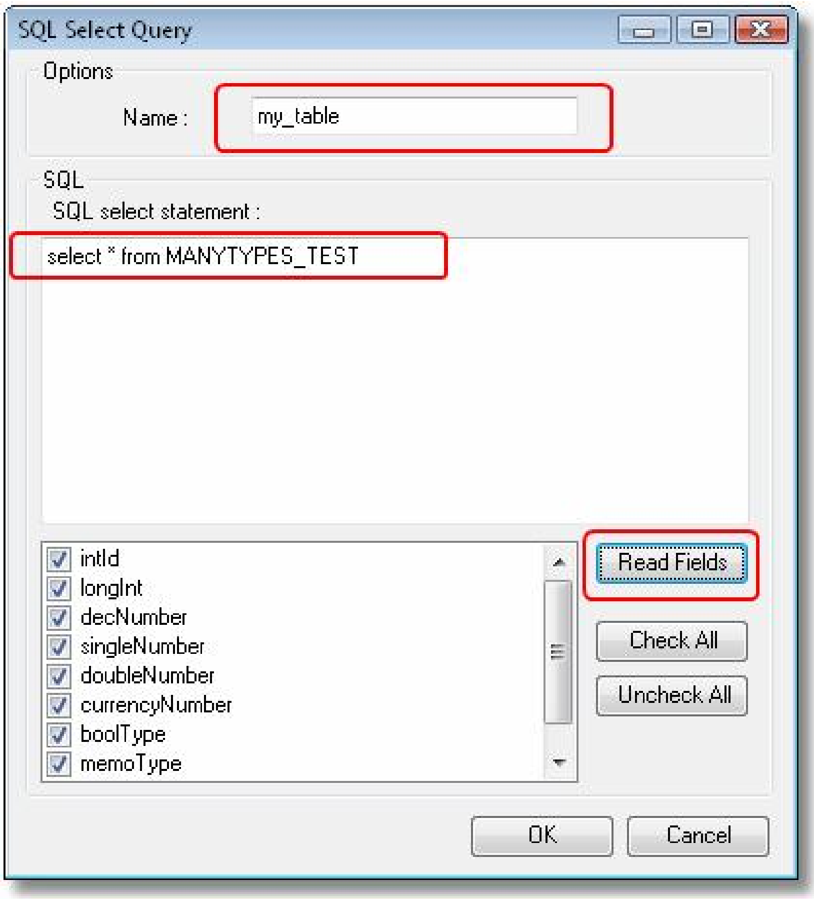
IniFile	
RootType	Input
SourceType	Odbc
SpecialInstruction	**None**
TranslationType	**Database**
Validation	
Validation	

Validation
Validation may be actual value expected in the item or XML Schema file path

Processing finished successfully

Adding SQL queries to database.

SQL queries during processing are executed in the order they are listed on the screen. Type query name and SQL statement in SQL Select Query dialog, then press “Read Fields” to get all the fields query can retrieve and press “OK”. Query and checked fields will be added to the map.



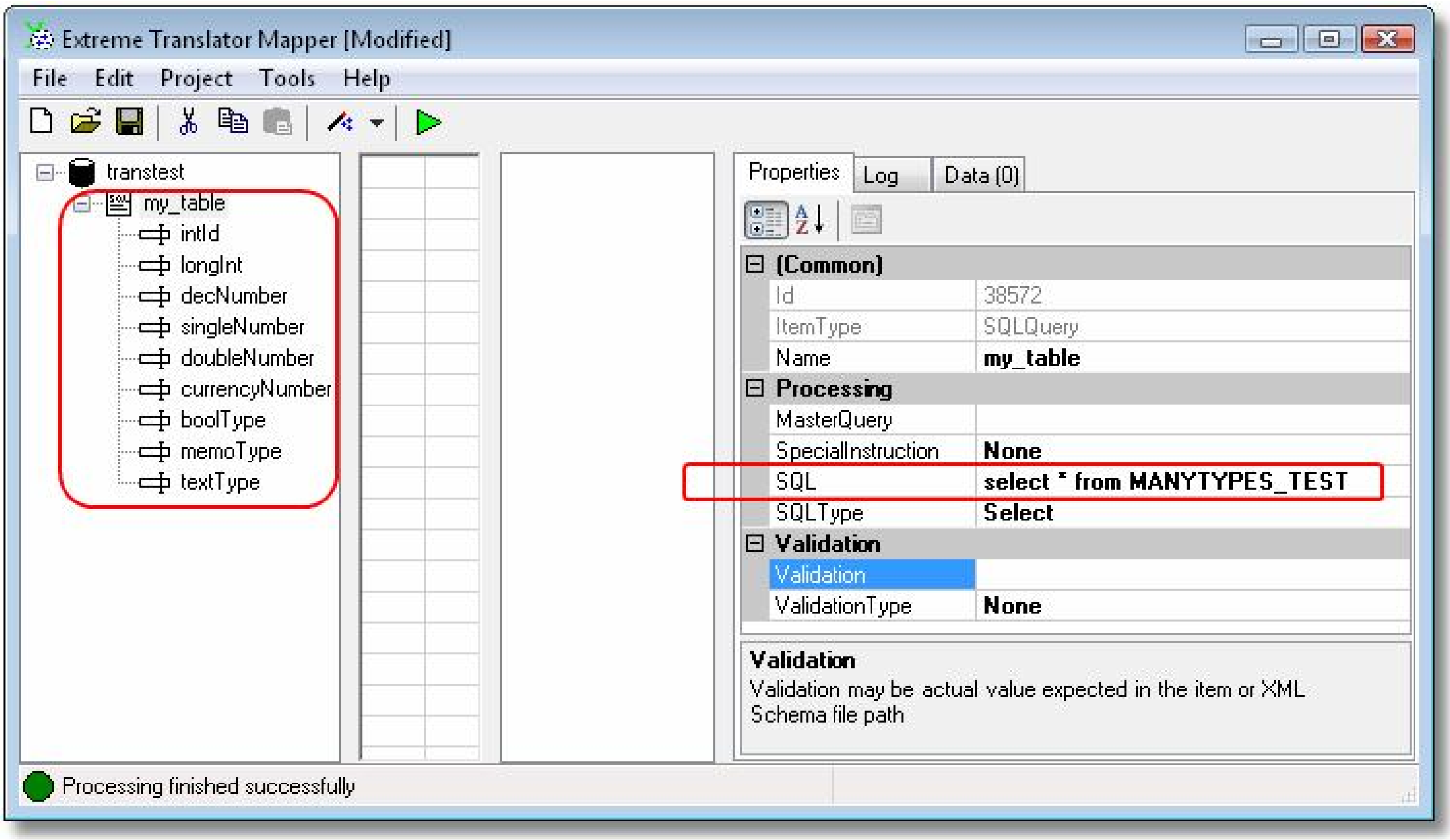
Name : my_table

```
select * from MANYTYPES_TEST
```

- [✓] intId
- [✓] longInt
- [✓] decNumber
- [✓] singleNumber
- [✓] doubleNumber
- [✓] currencyNumber
- [✓] boolType
- [✓] memoType

Adding query and fields to the map.

Map editor will try to detect queries field types and sizes. Some specific field types may not be detected properly and may result in data truncation or failure during data retrieval. SQL can be edited after query is added. If new fields are added to SQL statement (SQL property) and should be retrieved, you should add them under the query using “Add” and “Field” menu option, and map them to output.

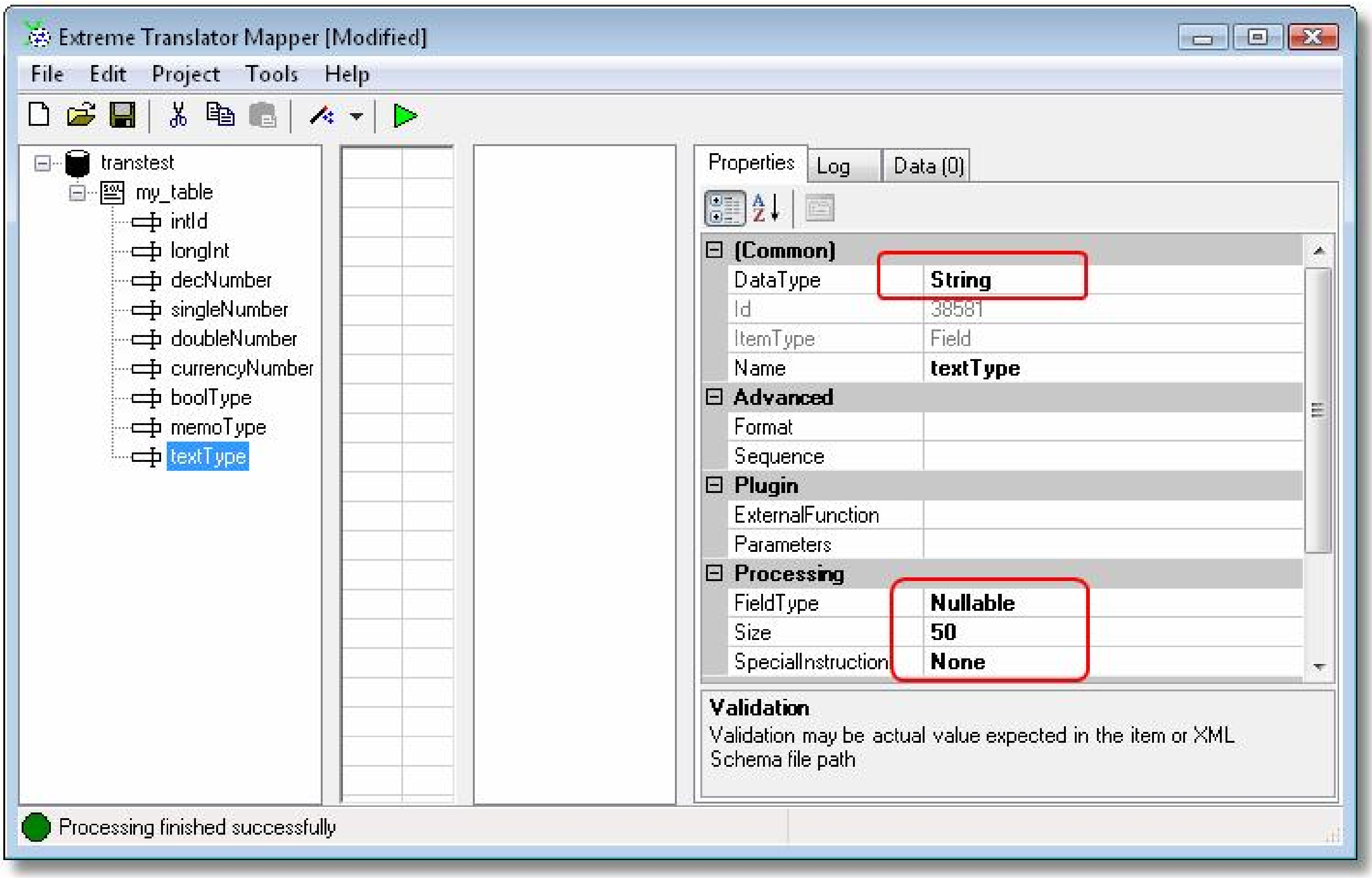


(Common)	
Id	38572
ItemType	SQLQuery
Name	**my_table**
Processing	
MasterQuery	
SpecialInstruction	**None**
SQL	**select * from MANYTYPES_TEST**
SQLType	**Select**
Validation	
Validation	
ValidationType	**None**

Validation
Validation may be actual value expected in the item or XML Schema file path

Processing finished successfully

Major query properties.


Extreme Translator Mapper [Modified]
File Edit Project Tools Help
Properties Log Data (0)

transtest
my_table
intId
longInt
decNumber
singleNumber
doubleNumber
currencyNumber
boolType
memoType
textType

(Common)	
DataType	String
Id	38581
ItemType	Field
Name	textType
Advanced	
Format	
Sequence	
Plugin	
ExternalFunction	
Parameters	
Processing	
FieldType	Nullable
Size	50
SpecialInstruction	None

Validation
Validation may be actual value expected in the item or XML Schema file path

Processing finished successfully

Major field properties.

Date, time and datetime fields accept data in a special format.

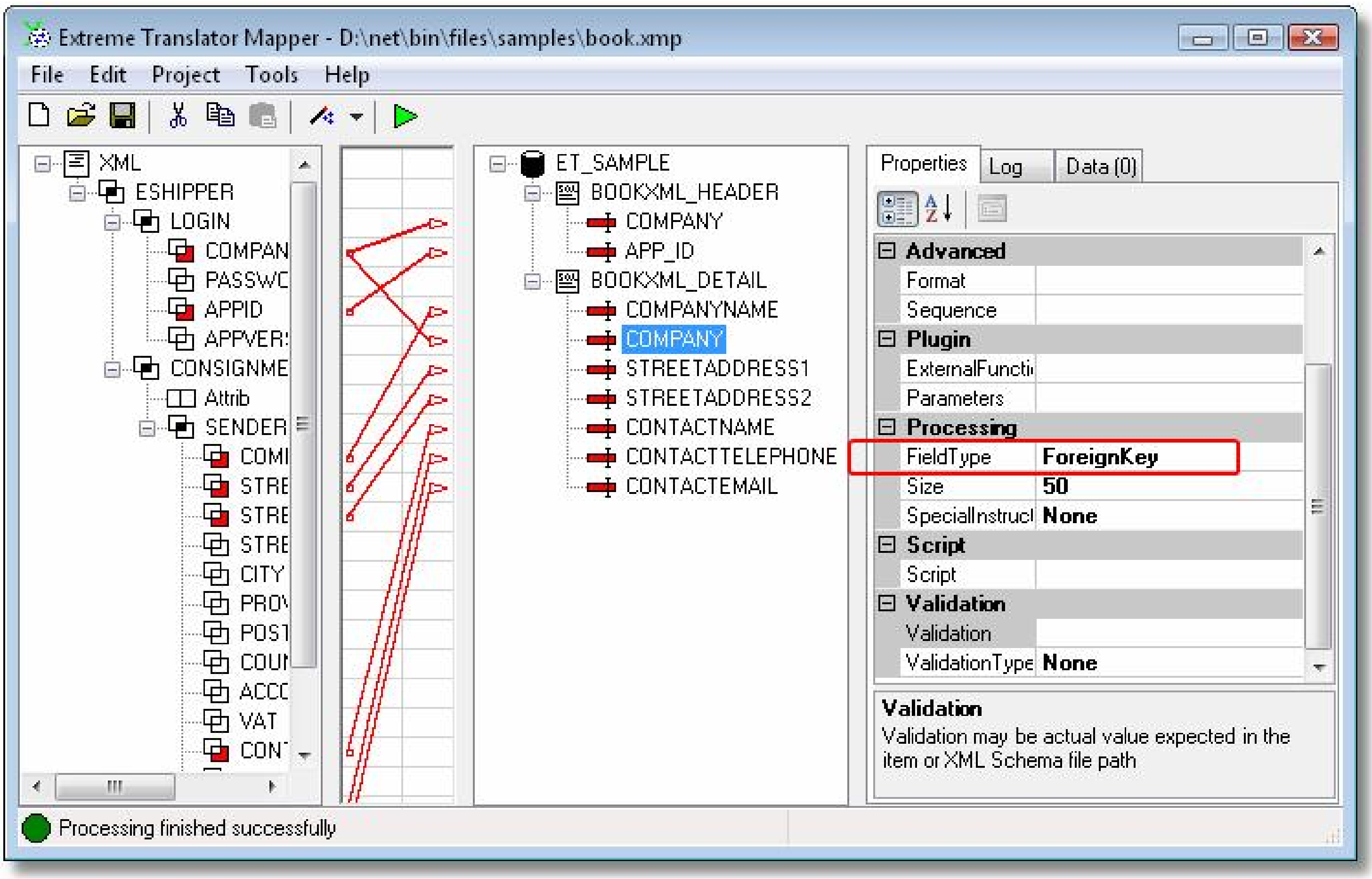
DataType property	Data format to be used
Date	yyyymmdd
Time	hhmmss
Datetime	yyyymmddhhmmss

Inserting data into database

Table has to have at least two fields for inserts to work.
FieldType property drives inserts. If you want to change the way records are inserted modify property for table fields. FieldType should match database field type in most cases. However there could be exceptions if you would like to reorder fields in some special way. You may have FieldType setup to NotNull or PrimaryKey even if actual field in the database table is Nullable.

Basic concepts built into the translator:
1. PrimaryKey field is like a leader field and should lead record ordering. There should be at least one PrimaryKey field in the table. All fields that come in the input stream before PrimaryKey should be marked as ForeignKey or Optional, and all fields that come in the input after PrimaryKey should be marked as NotNull or Nullable.
2. NotNull fields should always have data in them. If there is no data in the input, NotNull fields will “borrow” data from values in previous record.
3. Nullable fields can have they values discarded. If there is only one Key value and two Nullable field values, first value will be lost (overridden) with second value coming into Nullable field. Nullable values should come after PrimaryKey value in the input side.
4. Optional field is like Nullable field with only difference that it’s value can come from input data that is above first value that comes to PrimaryKey.
5. ForeignKey is used to mark field that is used to form relationship for PrimaryKey on other table.
Example: there are two tables Header and Detail. Header has PrimaryKey and some Nullable fields. Detail will have one

PrimaryKey, some Nullable fields and one ForeignKey. Detail ForeignKey value will come from the same item on input side where Header PrimaryKey data is coming from. Look into XML-to-Database sample for more details.
Using these concepts you should be able to achieve almost any record ordering.


Extreme Translator Mapper - D:\net\bin\files\samples\book.xmp
File
Edit
Project
Tools
Help
Properties
Log
Data (0)

XML
- ESHIPPER
  - LOGIN
    - COMPAN
    - PASSWC
    - APPID
    - APPVER:
  - CONSIGNME
    - Attrib
    - SENDER
      - COMI
      - STRE
      - STRE
      - STRE
      - CITY
      - PRO\
      - POST
      - COUI
      - ACCC
      - VAT
      - CON

ET_SAMPLE
- BOOKXML_HEADER
  - COMPANY
  - APP_ID
- BOOKXML_DETAIL
  - COMPANYNAME
  - COMPANY
  - STREETADDRESS1
  - STREETADDRESS2
  - CONTACTNAME
  - CONTACTTELEPHONE
  - CONTACTEMAIL

Advanced	
Format	
Sequence	
Plugin	
ExternalFuncti	
Parameters	
Processing	
FieldType	**ForeignKey**
Size	**50**
SpecialInstruct	**None**
Script	
Script	
Validation	
Validation	
ValidationType	**None**

Validation
Validation may be actual value expected in the item or XML Schema file path

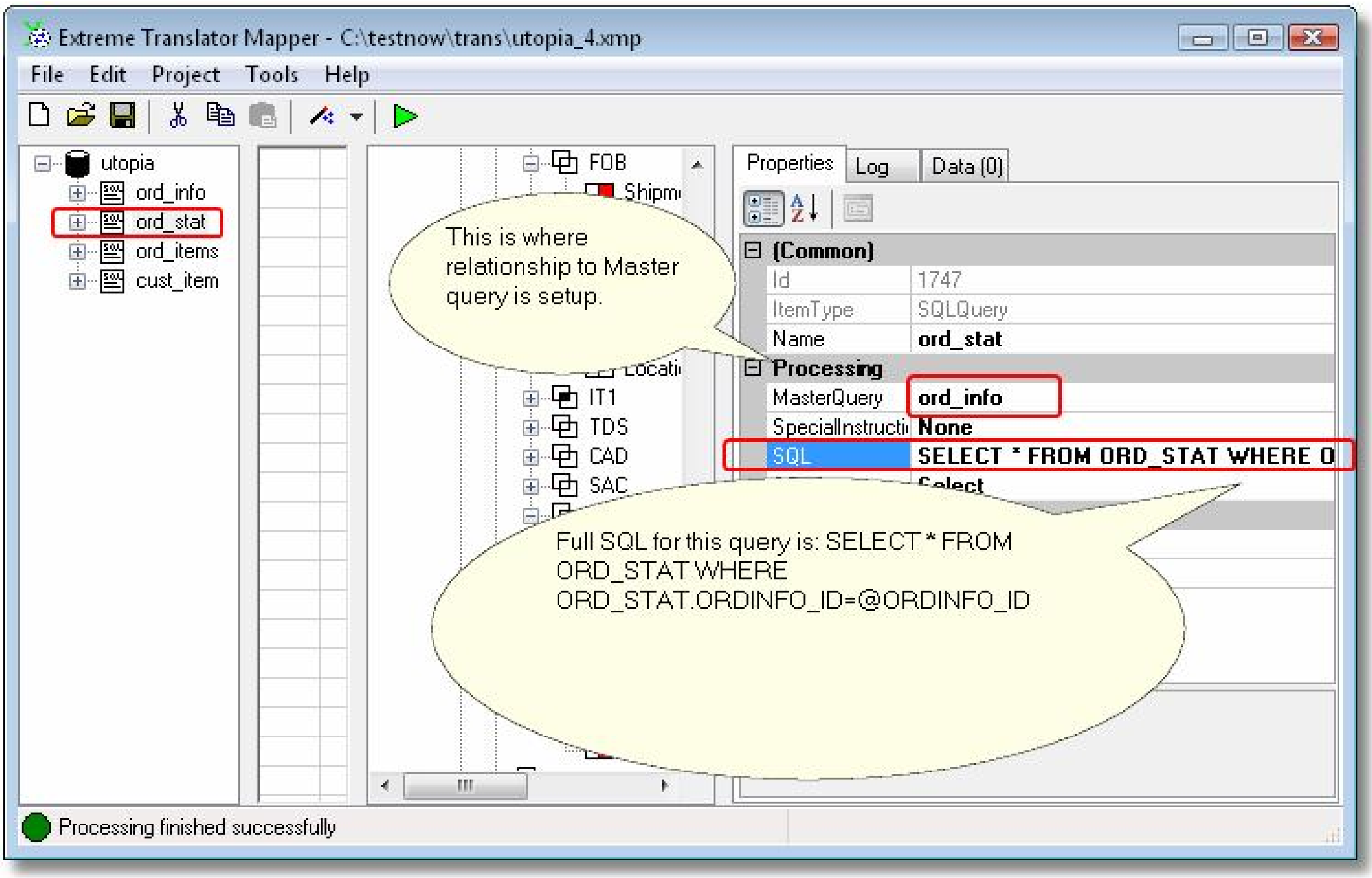
Processing finished successfully

FieldType property drives record ordering.

SQLType property for database table can be set to Insert, InsertOrUpdate or Update flag. Translator will use first PrimaryKey field in the database to perform Insert, InsertOrUpdate and Update operations. InsertOrUpdate is a combination of possible operations on the database where translator will select row from the database based on PrimaryKey, and if it exists translator will perform update otherwise insert will be performed.

Selecting data from database

Queries can be executed in sequence and retrieve results in master-detail layouts. In simplest scenario you would have two queries, one would provide data for header record, let say purchase order, and another one would provide data for detail record, say purchase order line item. One query would be something like “*SELECT* PO_ID, PO_AMOUNT *FROM* PR_ORDERS”, second one would be “*SELECT* PO_ID, PO_LINE_NO, PO_ITEM_NAME *FROM* PR_LINE”. Both queries can be tied using master-detail relationship. You can add both queries using “Add SQL Query” menu item and set MasterQuery property on PR_LINE to point to PR_ORDERS, also use field PO_ID to make a relation. Detail query should be modified to “*SELECT* PO_ID, PO_LINE_NO, PO_ITEM_NAME *FROM* PR_LINE PO_ID = @PO_ID”. Symbol “@” means that field from master query should be used to fetch results from this query.
Parameter names and field names are case sensitive. It is possible to nest queries 4 levels deep. That is header, detail1, detail2, trailer with each query passing parameters to the one nested below.

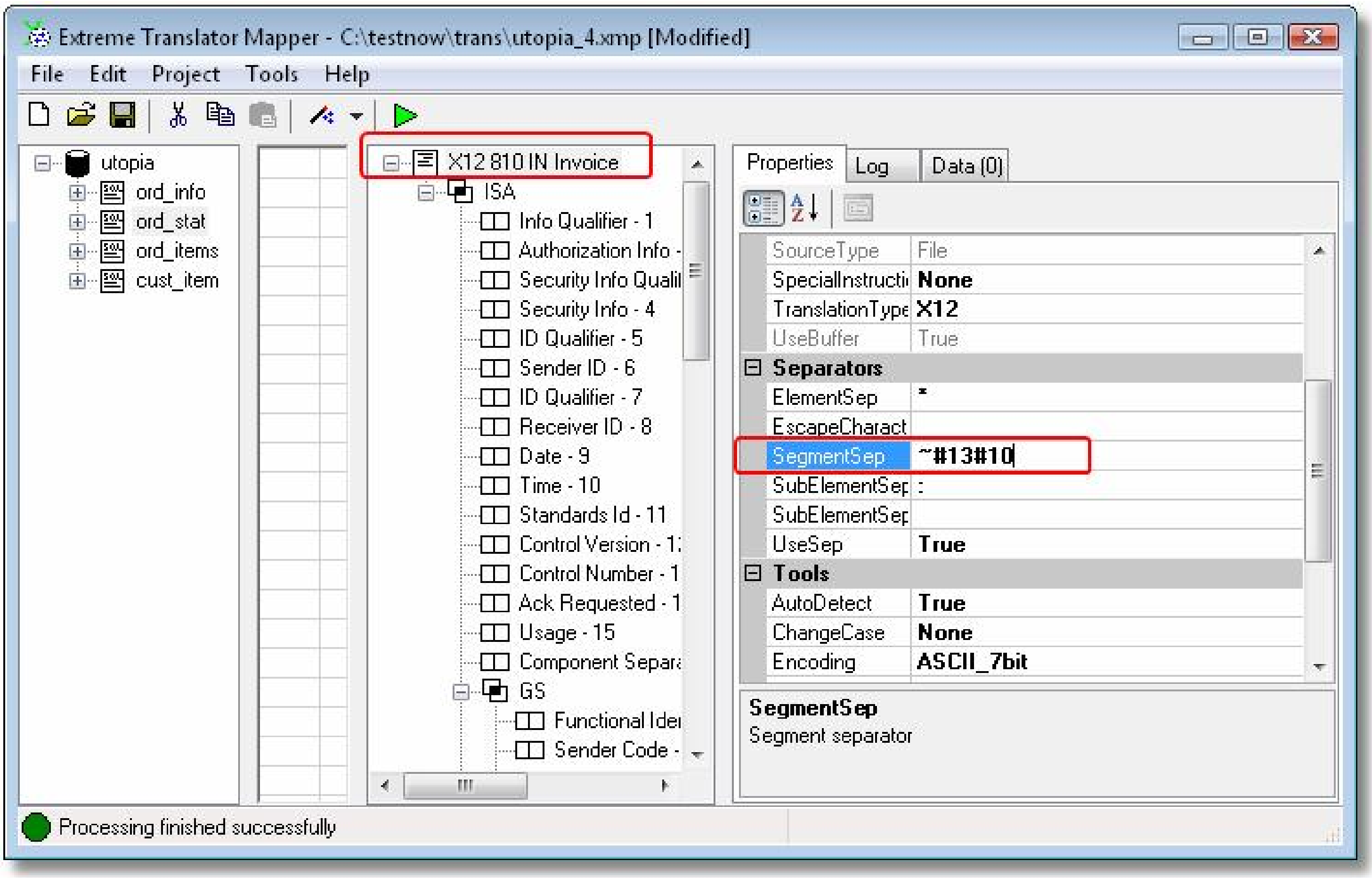

Extreme Translator Mapper - C:\testnow\trans\utopia_4.xmp
File Edit Project Tools Help
utopia
ord_info
ord_stat
ord_items
cust_item
FOB
IT1
TDS
CAD
SAC
Properties Log Data (0)
This is where relationship to Master query is setup.
Full SQL for this query is: SELECT * FROM ORD_STAT WHERE ORD_STAT.ORDINFO_ID=@ORDINFO_ID
Processing finished successfully

(Common)	
Id	1747
ItemType	SQLQuery
Name	ord_stat
Processing	
MasterQuery	ord_info
SpecialInstructi	None
SQL	SELECT * FROM ORD_STAT WHERE O

Example of detail query attached to the master. There ord_info is the MasterQuery while ord_stat is the detail. For each record of the master, detail is executed and ORDINFO_ID is passed into it.

Properties Tab

Properties Tab shows all the properties of the currently selected map item. You can edit any properties that are displayed in black. Property list can be displayed in groups: Common, Advanced, Separators, etc., or it can be sorted by property name. Some properties allow entering decimal codes in cases when they are not displayable characters and cannot be entered using the keyboard. One of most common cases is CRLF, carriage return and line feed, character sequence, it is also called new line. You can enter decimal characters #13#10 and they will be treated as CRLF. If your data is separated with just LF you can use #10 only. “#” should be entered before any decimal code. The properties where you can enter decimal codes are: *StartTag*, *EndTag*, *LoopEndTag*, *SegmentSep*, *ElementSep*, *SubElementSep, EscapeCharacter, Filter*.



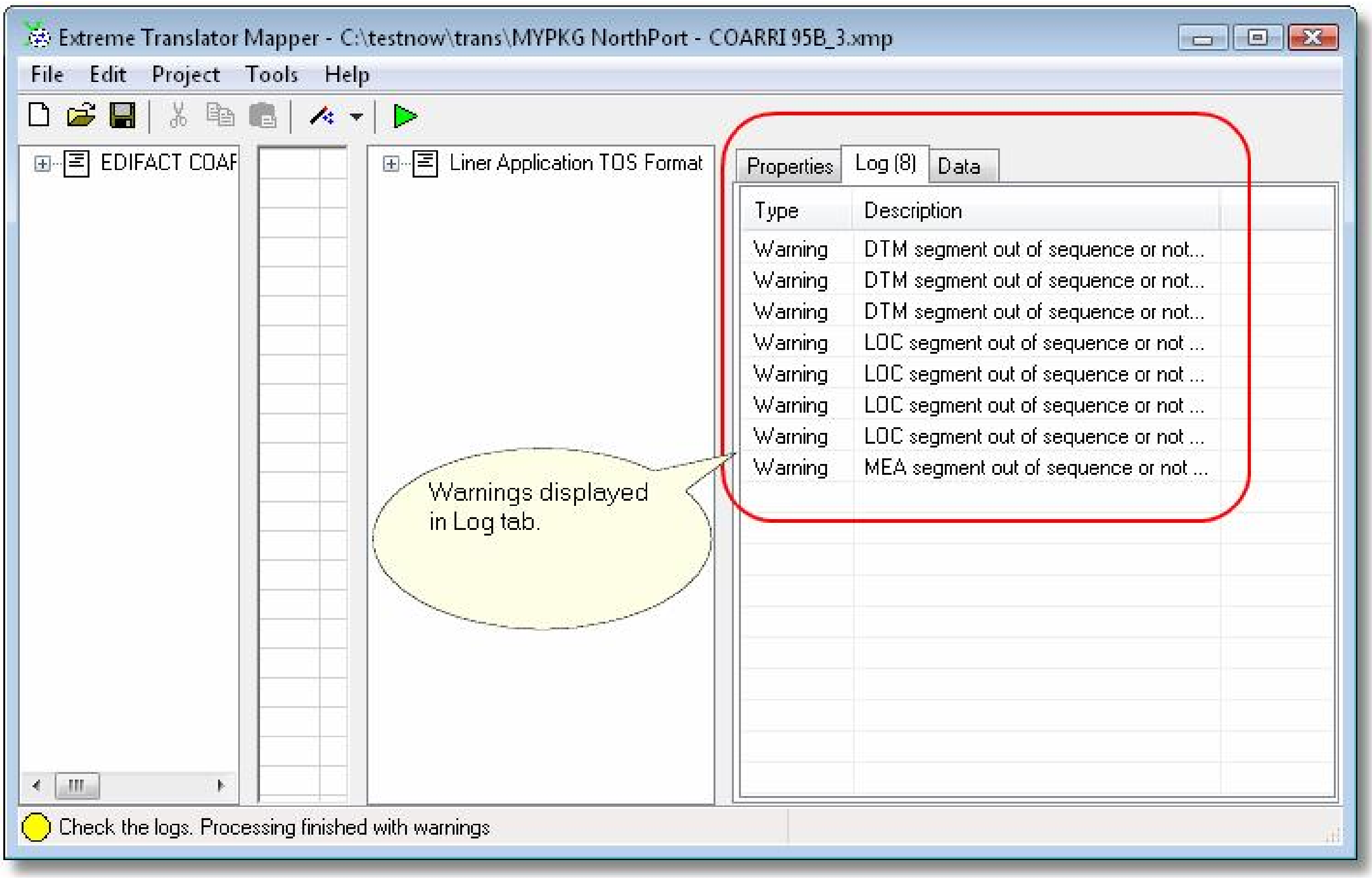
SourceType	File
SpecialInstructi	None
TranslationType	X12
UseBuffer	True
Separators	
ElementSep	*
EscapeCharact	
SegmentSep	~#13#10
SubElementSep	:
SubElementSep	
UseSep	True
Tools	
AutoDetect	True
ChangeCase	None
Encoding	ASCII_7bit

SegmentSep
Segment separator

There is how to use non printable characters in properties. In this example carriage return and line feed is placed in the output file.

Log Tab

Log Tab should display warnings and non-critical errors in a form of the list. That means processing finished with warnings and minor errors but output was produced anyway. Most of warnings are data validation errors. Some of them are based on assumption that data we received do not match certain rules defined in the map however those restrictions are not critical and output was produced anyway.


Extreme Translator Mapper - C:\testnow\trans\MYPKG NorthPort - COARRI 95B_3.xmp
File Edit Project Tools Help
EDIFACT COAF
Liner Application TOS Format
Properties
Log (8)
Data
Warnings displayed in Log tab.

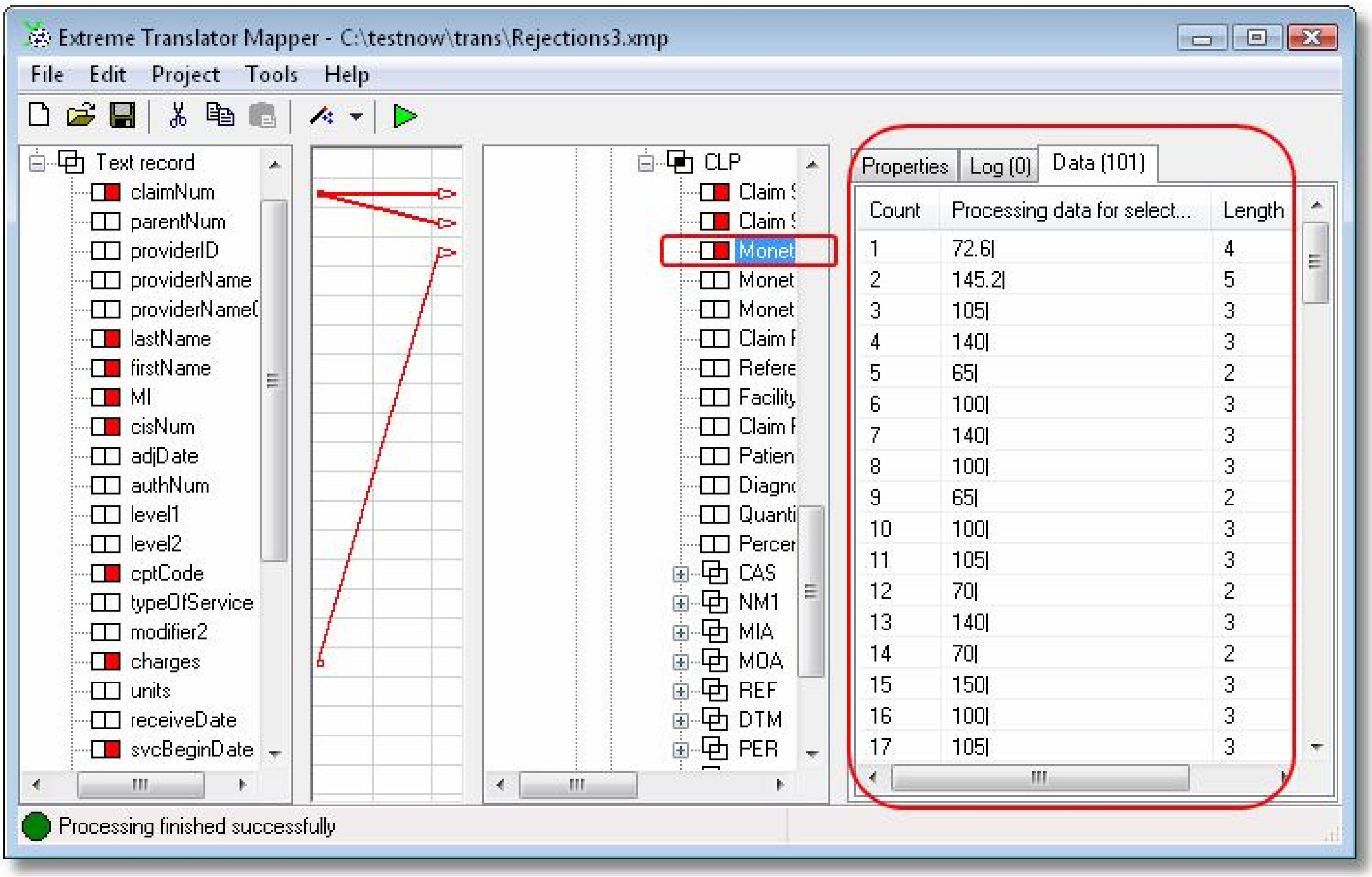
Type	Description
Warning	DTM segment out of sequence or not...
Warning	DTM segment out of sequence or not...
Warning	DTM segment out of sequence or not...
Warning	LOC segment out of sequence or not ...
Warning	LOC segment out of sequence or not ...
Warning	LOC segment out of sequence or not ...
Warning	LOC segment out of sequence or not ...
Warning	MEA segment out of sequence or not ...

Check the logs. Processing finished with warnings

Warnings produced during execution.

Data Tab

Is designed to show processing data attached to the map item. After processing each item in the map has data associated with it. That is whatever was retrieved from the file or written to the file and attached to the map item. You can select item and see data attached to it. Each item can have 0-to-many data blocks attached to it. This is because most of the map items can loop.



Count	Processing data for select...	Length
1	72.6|	4
2	145.2|	5
3	105|	3
4	140|	3
5	65|	2
6	100|	3
7	140|	3
8	100|	3
9	65|	2
10	100|	3
11	105|	3
12	70|	2
13	140|	3
14	70|	2
15	150|	3
16	100|	3
17	105|	3

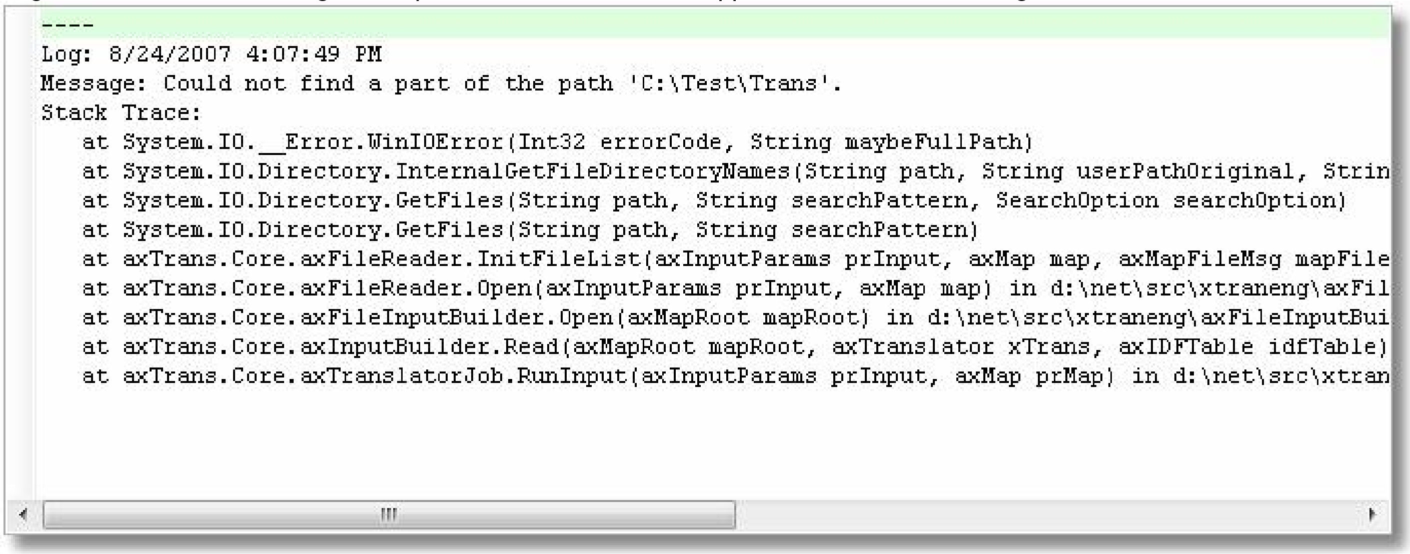
Processing finished successfully

Data attached to and processed for each map item after execution. Data tab also shows items length and marks end of item with "|". It is handy if item has spaces at the end of data so you can see them.

Technical Support

When critical errors happen in the code during processing of the map, log file is produced. It contains instructions and function names that failed. This log may not be useful for the end user but is excellent source of information for support.

Log file name starts with original map file name where error happened and ends with ".log".



```
----
Log: 8/24/2007 4:07:49 PM
Message: Could not find a part of the path 'C:\Test\Trans'.
Stack Trace:
   at System.IO.__Error.WinIOError(Int32 errorCode, String maybeFullPath)
   at System.IO.Directory.InternalGetFileDirectoryNames(String path, String userPathOriginal, Strin
   at System.IO.Directory.GetFiles(String path, String searchPattern, SearchOption searchOption)
   at System.IO.Directory.GetFiles(String path, String searchPattern)
   at axTrans.Core.axFileReader.InitFileList(axInputParams prInput, axMap map, axMapFileMsg mapFile
   at axTrans.Core.axFileReader.Open(axInputParams prInput, axMap map) in d:\net\src\xtraneng\axFil
   at axTrans.Core.axFileInputBuilder.Open(axMapRoot mapRoot) in d:\net\src\xtraneng\axFileInputBui
   at axTrans.Core.axInputBuilder.Read(axMapRoot mapRoot, axTranslator xTrans, axIDFTable idfTable)
   at axTrans.Core.axTranslatorJob.RunInput(axInputParams prInput, axMap prMap) in d:\net\src\xtran
```

There are sample log file contents.

Please visit “Support” page on our website for more information.