

# Cross Translator

## Cross Translator v1.2x User Manual

Copyright © 1998-2007 Etasoft Inc.

Main website <http://www.etasoft.com>

Cross Translator website <http://www.ctranslator.com>

Purpose.....	3
Additional Documentation.....	3
Requirements.....	3
License.....	3
Using samples.....	3
Terminology.....	4
Software package.....	5
The role of translators.....	5
What translators can do?.....	6
What are the maps?.....	6
Map Editor.....	6
How to do the mapping?.....	7
Integration.....	8
Plug-ins and Java.....	8
Options.....	9
Batch processing.....	11
Dialog based processing.....	12
Transactions.....	13
Properties.....	13
DataPath property.....	15
Script property.....	16
Function property.....	17
Condition property.....	17
Validation property.....	18
Templates.....	19
Translation.....	20
EDI X12, EDIFACT translation.....	21

HI PAA support ..... 22

XML translation ..... 22

XML schema validation..... 23

Flat file translation..... 23

**Flat text file with delimiters** .....23

**Flat text file fixed length**.....25

Database mappings..... 26

**Basics**.....26

**Master detail queries**.....29

**PrimaryKey and ForeignKey fields** .....30

Log tab ..... 31

Processing tab..... 32

Data tab ..... 33

Performance Tuning ..... 34

Technical Support..... 35

Appendix A. List of build-in plug-ins..... 37

## Purpose

Cross Translator is cross platform data translation and transformation software package. It features lightweight and easy to use Map Editor, open architecture for external plug-ins, transactional processing and much more. It is written in Java™ language for better portability.

## Additional Documentation

User's Manual describes basic use of the Map Editor and translator utilities. Additional documentation that covers specifics of X12/EDIFACT, XML, flat file translations, plug-ins and Developer SDK is already copied into installation directory or can be downloaded from the website.

## Requirements

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

CPU	Pentium 4, 2.0 GHz
RAM	512Mbt
HDD	5Gbt
Operating System	OS with Java Runtime v1.4

If you are working with big data files and map files that are more than 200Kbt in size you need much faster machine. This is recommended configuration:

CPU	Pentium 4, 2.8 GHz
RAM	512Mbt
HDD	5Gbt
Operating System	OS with Java Runtime v1.4

## License

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.

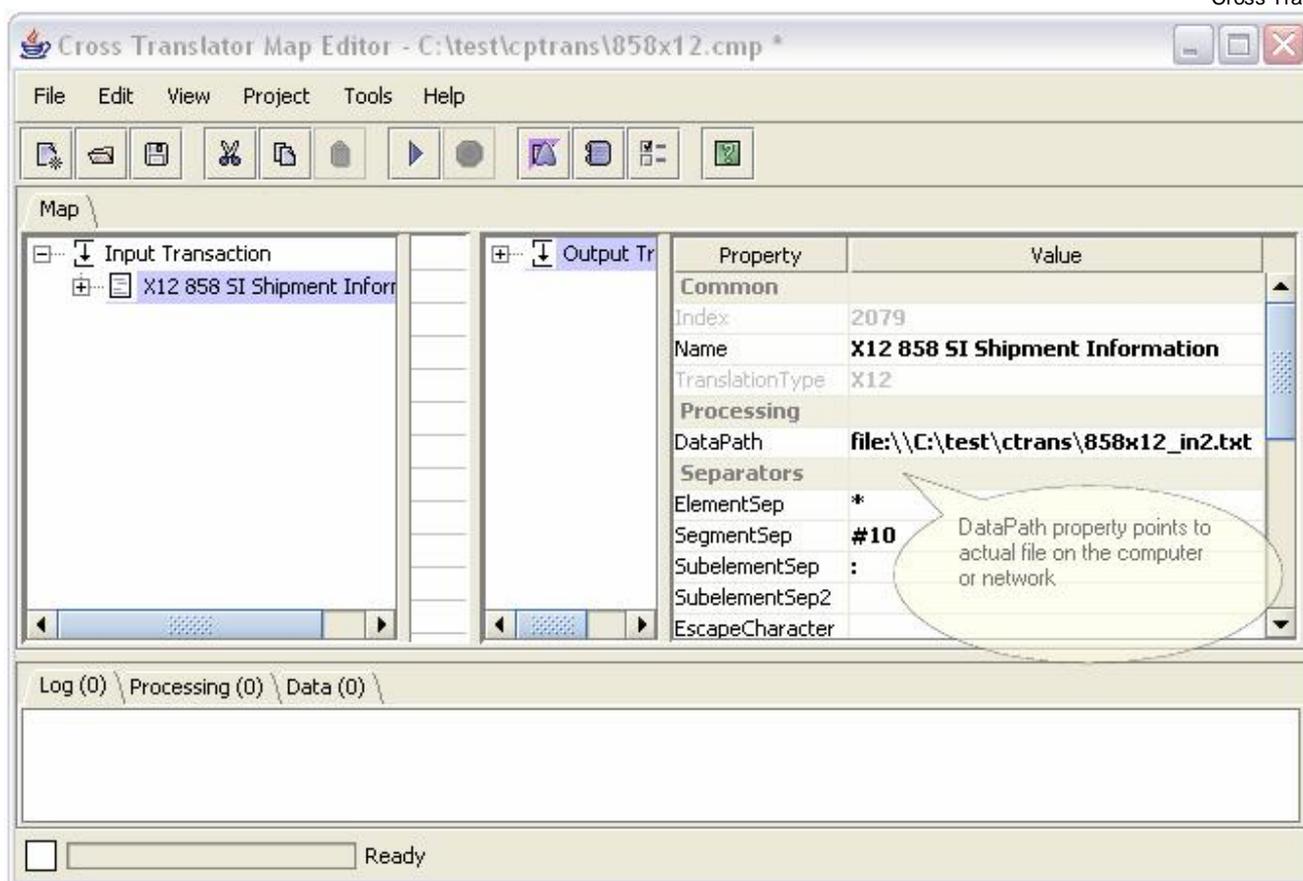
You can register and buy licenses online. After the purchase email will be send to you with the license key.



Enter your license key in Product License screen.

## Using samples

Files with extension .cmp can be opened using Map Editor. They are map files used to define translation rules and mappings. Each map has a set of messages defined. They are items just below Input Transaction or Output Transaction. Each message has DataPath property. This property points to sample file in directory C:\test\cprtrans\_cases\. Full DataPath includes prefix in a form of "file://".



You can choose to copy files to `C:\test\cptrans_cases\` directory or change it to your input and output file names. Sample maps are provided for demonstration purposes. They are not complete maps. Complete maps cannot be created as organizations and companies have different requirements for input and output files and how they should be laid out. That is main reason behind Map Editor tool and over 2000 templates that come with it. Easy customization is the key.

List of sample maps provided:

Map File	Description	Data Files
837text.cmp	EDI X12 837 translation to flat file	Just a map, use your X12 837
858text.cmp	EDI X12 858 translation to flat file	Just a map, use your X12 858
Desadvtxt.cmp	EDIFACT DESADV translation to flat file	Just a map, use your DESADV
Recadv.cmp	Flat text file translation to EDIFACT RECADV	Included
txtrecadv.cmp	Flat text file translation to XML	Included
xmlrecadv.cmp	XML file translation to EDIFACT RECADV	Included
850xml.cmp	XML file translation to EDI 850	Included

## Terminology

Translator uses special definitions to unify mapping similarities between EDI X12, EDIFACT, XML, flat files and other formats. Definitions used by mapping tool have slightly different meaning depending on the data translation type.

List of definitions and they explanations:

Definition	Translation Type	Meaning
Message	All translation types	File or data stream to be processed.
Segment	EDI X12, EDIFACT	EDI segment, such as ISA, GE, ST or UNB, UNH, UNT
	XML	XML tag, such as <header></header> or <order></order>

	Flat file	Block of text, such as line of text in flat file, could be line of text terminated with carriage return and line feed in comma separated value (CSV) file
Element	EDI X12, EDIFACT	EDI element, block of data inside of segment, terminated by EDI separator
	XML	XML attribute, such as name below: <user name="some_name"/>
	Flat file	Block of text inside of segment, such as text separated by commas in comma separated value (CSV) file
Subelement	EDI X12, EDIFACT	EDI component sub element, block of data inside of element
	XML	Not used
	Flat file	Not used

## Software package

There are two software packages available for download:

1. Package for Windows based systems
2. Package for Unix and Linux based systems.

First package is oriented towards translation map development and translation testing. It comes with sample Java™ code for plug-ins and application integration. Map Editor is major component of this package. All the applications are post-compiled into Windows executables (EXE). However you can also run those tools using JAR files from integration directory.

Second package is oriented towards server installation. It features small size and only most important files are included in this package.

In case if you would like to run translator tools via Java™ environment you can use JAR files provided from any one of the packages.

Next line will run Map Editor:

```
>java -classpath [list all JAR files here] axMapEditor
```

Next line will run Map Runner:

```
>java -classpath [list all JAR files here] axMapRunner
```

Next line will run command line processing for the map:

```
>java -classpath [list all JAR files here] axBatchRun [your map file] [other options]
```

Command line processing can execute as part of more complex script or process.

**IMPORTANT:** if ctrancmd.jar is not included in the /integration/lib directory you can request it via email to technical support. This is due to source code privacy. We want to protect our Java source code from illegal use. ctrancmd.jar is only required if you want to run translator via Java™ environment.

## 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:

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. However you do not have full control on how translation is performed. Also software integration options are limited.

## 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.

## What are the maps?

Translator maps are sets of data translation rules defined using graphical mapping tool and saved in \*.cmp 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. 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 cannot be simply defined by changing properties. In order to solve these problems we have introduced scripting support and plug-ins.

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 use 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.

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.

## Map Editor

Map editor is divided into two main panes: Map and Log. Map pane is also divided into four additional 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". 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.

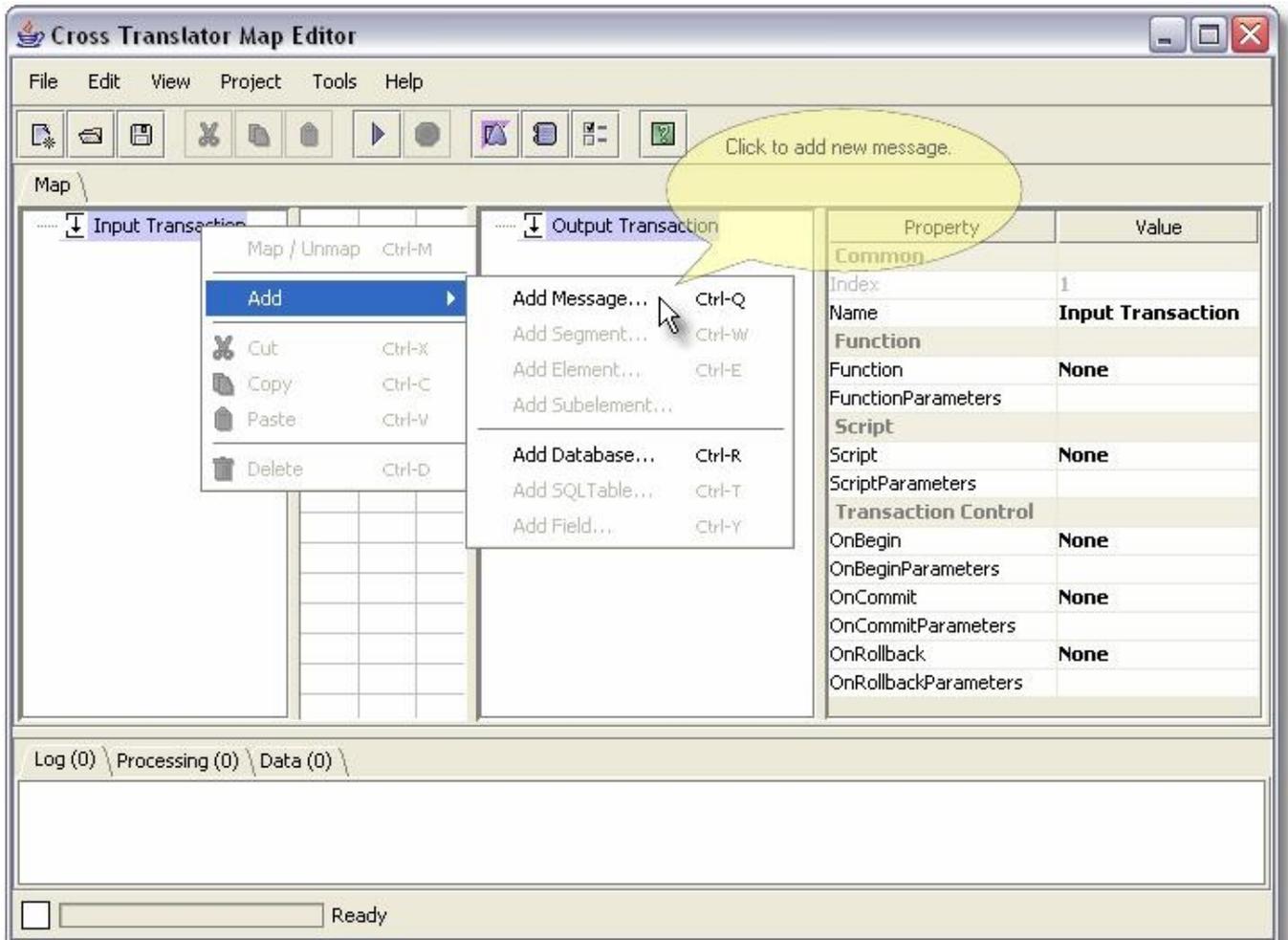
Simplest way to add some items is to use "Add..." menu item, however there are some limitations in order what can be added to what.

For message based mapping:

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

For database based mapping:

1. Only tables or queries can be added to the database.
2. Only fields can be added to tables and queries.
3. Nothing can be added to fields.

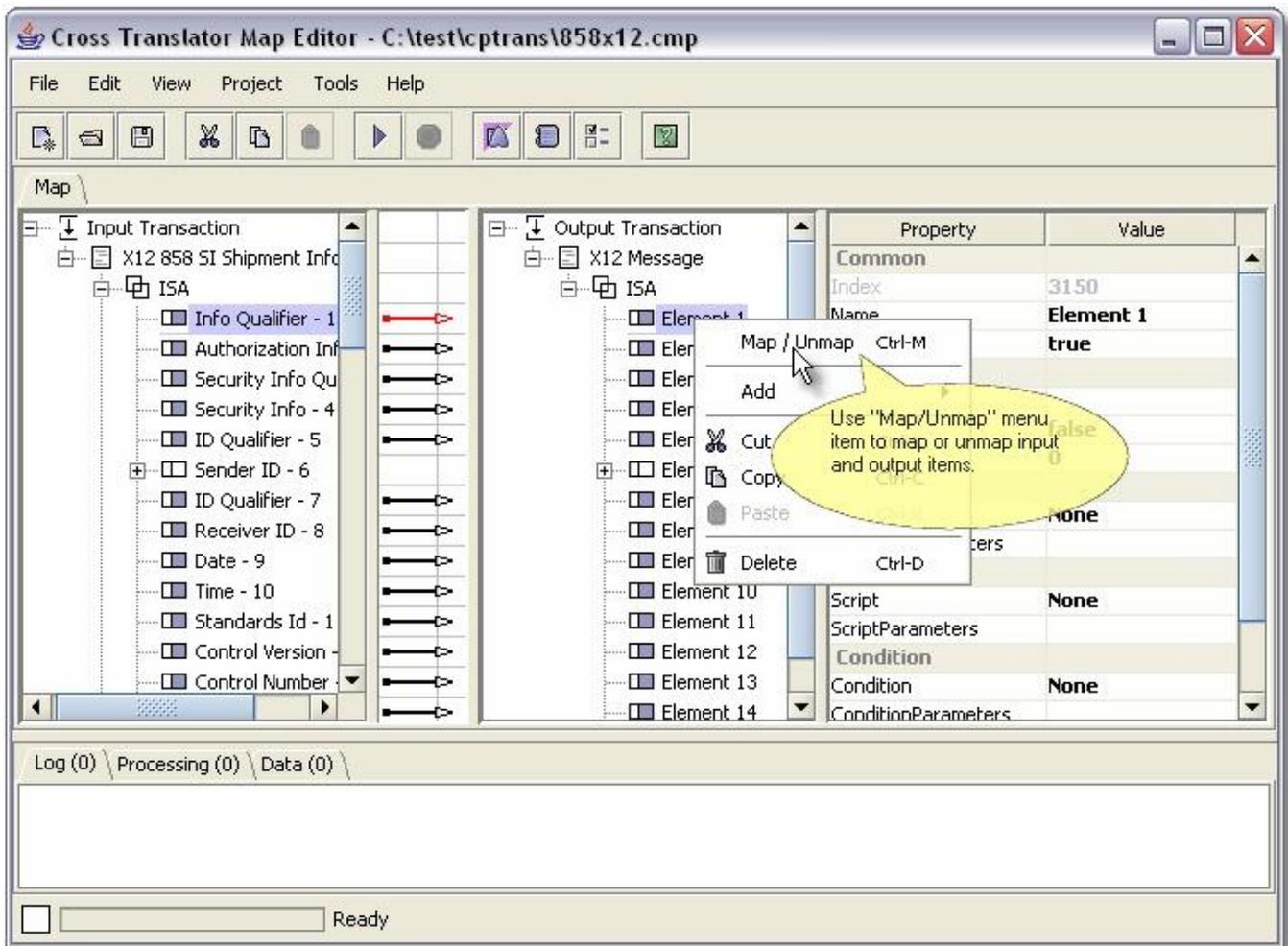


Adding message to input transaction by using right click popup menu.

### How to do the mapping?

Once input and output sides are defined, you can create mappings across. They are displayed as arrows in second pane in the Map Editor.

Mapping is done via main or popup menu when both items in both mapping trees are selected.



This is how mapping can be done using popup menu on the right (output) mapping tree.

## Integration

If you already have some business applications and would like to integrate your software with Cross Translator there are two types of scenarios for integration:

1. Internal process calls
2. External process calls

Internal calls can be made via Cross Translator Developer SDK in Java™. They are direct calls to translator engine. They let you load maps, execute them and analyze warnings and errors. It is most effective way of integration. Cross Translator is coded in Java™ but for use on Windows platform it is packaged into executable package (EXE). Special documentation is provided on how to perform integration using Java™.

If your business applications are not Java™ based, or you have no source code for them, internal integration scenario may not be possible. External calls can be made via command line batch run utility. They are less efficient but can be executed via user shell, scheduling tools (also available on Unix systems, tools like cron), also called from other applications.

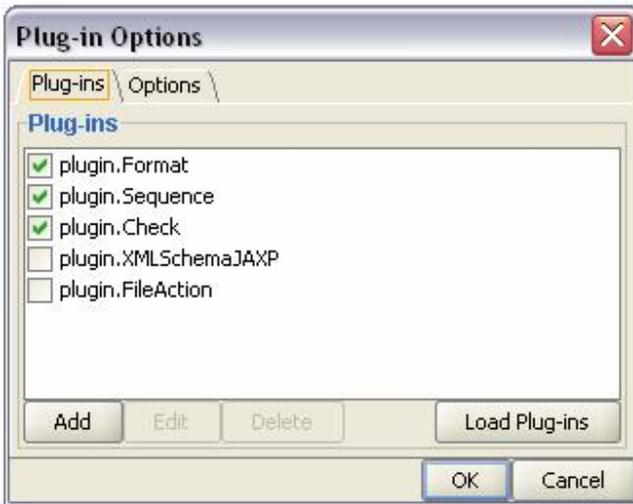
## Plug-ins and Java

Plug-ins are external functions called by translator during execution. Some plug-ins come with translator libraries in standard installation package, but even so they are still thought of as external functions and classes, and calls translator makes to them are performed same way as they would be external functions.

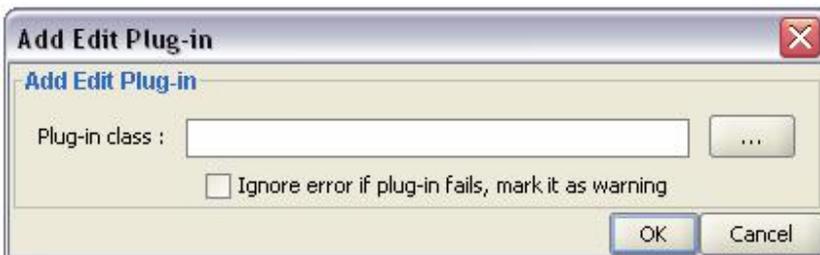
Plug-ins can be developed in-house and added to translator engine and defined as external functions in the map. They should be defined in "Plug-in Options" and CLASSPATH variable once translator starts. If you are using translator compiled into executable (EXE) then you can use CT\_PLUGINS environment variable to hold path that would otherwise would be set in CLASSPATH. CT\_PLUGINS acts as CLASSPATH variable for translator but it could be better option if you do not want to pollute your global CLASSPATH.



New plug-ins can be added and plug-in options can be changed.



"Plug-in Options" lists all available registered plug-ins. Plug-ins with check mark are enabled and can be used in the Map Editor. Once enabled they functions will be loaded and available in Function, Condition and Validation properties.

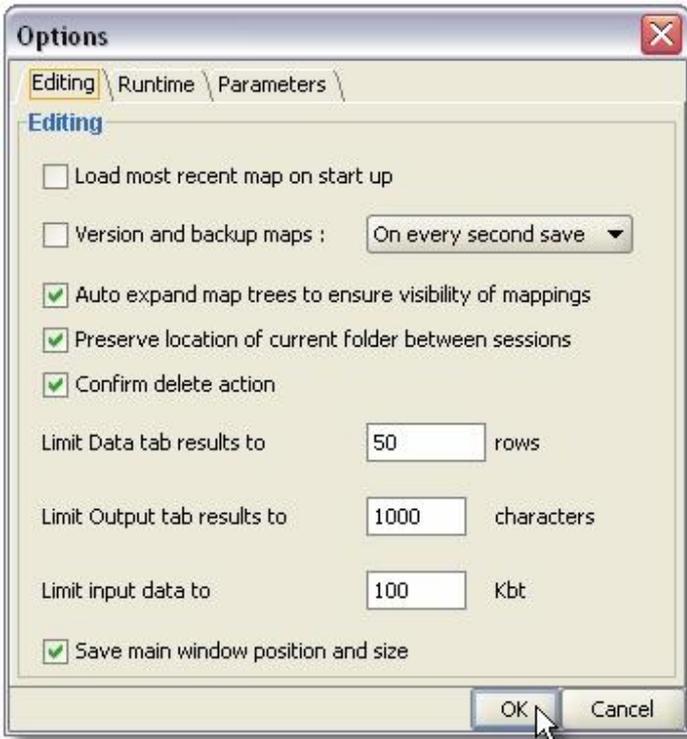


Plug-in class should not include extension ".class", and should be in the form package\_name.class\_name.

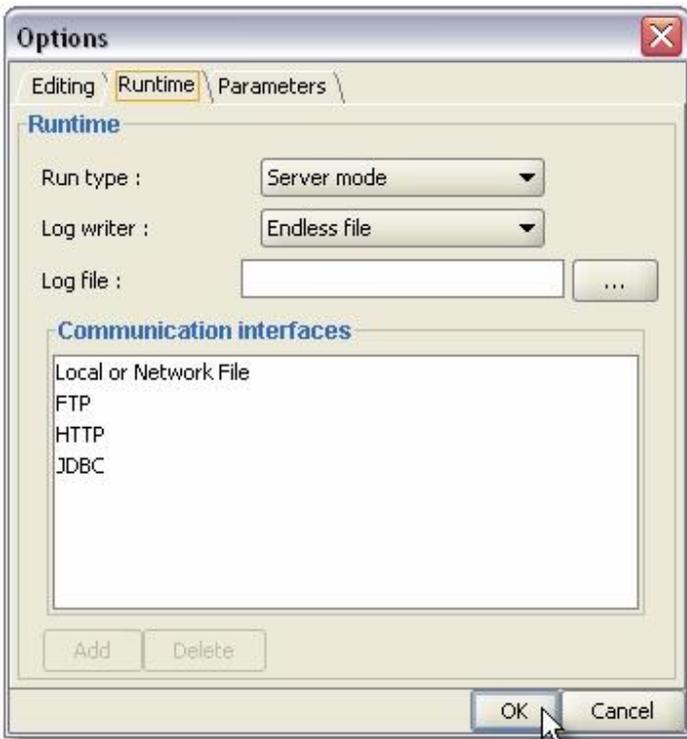
## Options

You can setup Map Editor to remember window sizes, most recent directory, load most recent maps on start up, show confirmation message upon delete and much more.

"Version and backup maps" is useful when saving maps over slow connections to the network drives. Map Editor will make backup copies of the map adding "~" at the end. In case if you would like to rollback changes and restore previous older maps all you need to do is remove "~" from the end of file name and load that map into Map Editor.



Editing options help you enable and disable most basic Map Editor features.



Map runtime options used by translator engine during execution.

Translator engine uses different thread priorities when "Run Type" is set to "Server mode" and when it is set to "Client mode". "Server mode" assumes that this is more powerful server machine and translator can run with higher priority. "Log Writer" defaults to "Endless file", but future versions of translator will support other log writer types. You can setup log file to be used by translator when map is executed via Map Editor in "Log file". If log file name is not supplied translator will take map file name and add extension ".log" to it. Example: if your map file name is

“myxmlmap.cmp” then your log file name will default to “myxmlmap.cmp.log”. If you just started with the mapping and map is not saved in the Map Editor into a \*.cmp file, log will default to file “logfile.log”.

This version of translator supports fixed set of communication protocols, such as file://, ftp://, http:// and jdbc://. They cannot be changed and no new interfaces can be added in current version.



Extra parameters can be passed to translator engine via “Parameters” tab. Behind the scenes Map Editor passes some default parameters to translator engine. Those are the parameters passed by default:

1. LICENSE – license number defined in “Product License” screen.
2. PRIORITY – priority can take values MAX or MIN. MAX stands for “Server mode” processing with translator engine running at high priority.
3. LOG – log file name.

You do not need to add those parameters.

## Batch processing

Maps can be executed via batch processing in the command line. You can use ctrancmd.exe on Windows. There are number of parameters that can be passed into batch map processing utility. Such as:

1. LICENSE – license number defined in “Product License” screen in Map Editor.
2. PRIORITY – priority can take values MAX or MIN. MAX stands for “Server mode” processing with translator engine running at high priority.
3. LOG – log file name.

First parameter must be map file name. This parameter is mandatory. You should also pass LICENSE key. It can be found under Map Editors “Product License” menu.

```

C:\> D:\Java\optrans\bin>ctran C:\test\ctrans\858x12.cmp License=TTTT-FUUU-L304-X6FU
Cross Translator Batch processing utility
Copyright (C) 2004-2005 Etasoft Inc.

LOG file name is: log.txt
Started at: 10/9/04 7:02 PM
Started processing
Started transaction Input Transaction
Begin Transaction: Input Transaction
Committed transaction Input Transaction
Commit Transaction: Input Transaction
Started transaction Output Transaction
Begin Transaction: Output Transaction
Committed transaction Output Transaction
Commit Transaction: Output Transaction
Completed processing

-----
Finished in: 1 second(s).
COMPLETE

D:\Java\optrans\bin>
D:\Java\optrans\bin>

```

Map processing via command line.

Map object properties can be changed during runtime using special command line parameters. For example if you want to change DataPath property to point to other outgoing file you can use parameter string below:

```
ID:3435.DataPath=file://C:\mydirectory\mynewfile.txt
```

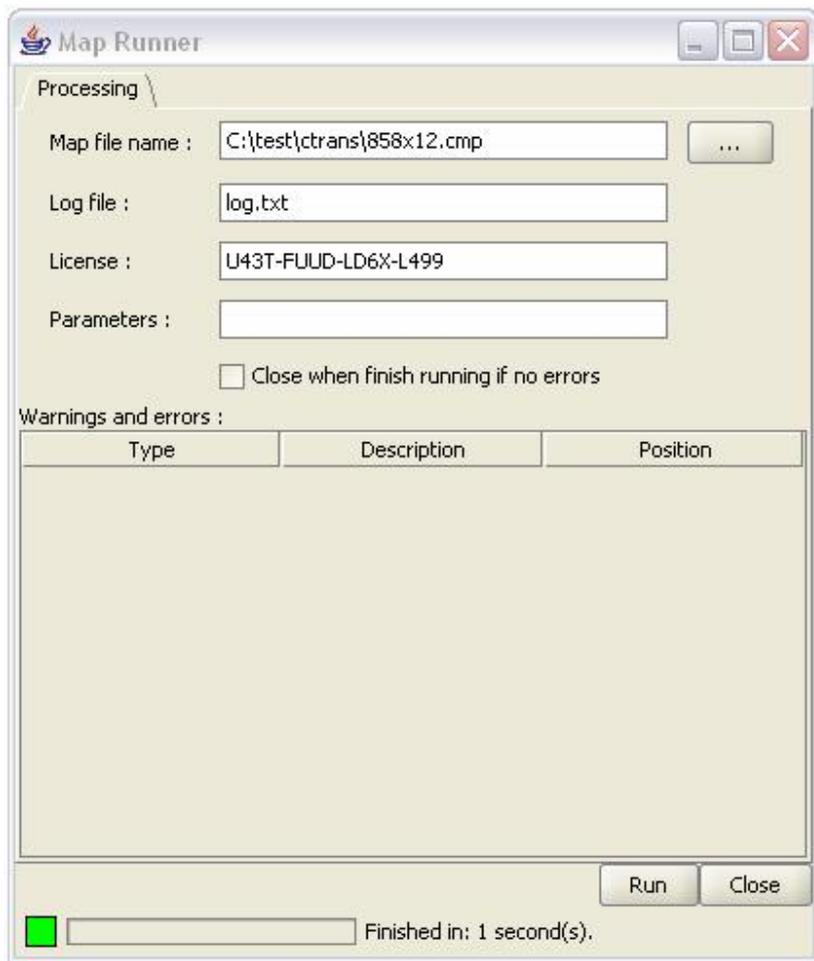
This expression above means: find map object with Index 3435 and assign new value to the property DataPath. There is another example on how to change property at runtime using command line parameters. In this case it is segment separator that we change:

```
id:2345.SegmentSep=~
```

String, numeric and boolean (true/false) property values can be passed in via command line. Property values that appear in drop down lists cannot be passed in. For example: plug-in names in Function, Condition and Validation property values cannot be overwritten via command line.

## Dialog based processing

Dialog based processing takes same parameters as command line processing, and you can also pass some additional parameters via Parameters property.



Dialog based processing can be used to run maps created with Map Editor manually. There are no mappings displayed so there is no danger that map would be modified accidentally, corrupted and saved.

## Transactions

Translator processing logic was built around notion of transaction as indivisible unit of work. There are three transactions involved in processing of the map:

1. Input transaction
2. Output transaction
3. Global transaction

Input and Output transactions are surrounded by Global transaction.

Transactions go through stages. They begin, finish successfully (commit) or fail (rollback). You can attach special functions to handle OnBegin, OnCommit and OnRollback events in the Map Editor. This mechanism can help handle all kinds of scenarios. For example translation error handling: OnRollback translator could call external plug-in to send email to notify system administrator that translation processing failed and processing has to restart with human intervention.

## Properties

Each map object on the screen has its own properties. Some properties are common to more than one item.

Property	Value
<b>Common</b>	
Index	3176
Name	<b>X12 Message</b>
TranslationType	X12
<b>Processing</b>	
DataPath	<b>file://C:\test\ctrans\858x12_out2.txt</b>
<b>Separators</b>	
ElementSep	*
SegmentSep	~#13#10
SubelementSep	:
SubelementSep2	
EscapeCharacter	
UseSeparators	<b>true</b>
AutoDetect	<b>true</b>
<b>Function</b>	
Function	None
FunctionParam...	
<b>Script</b>	
Script	None
ScriptParameters	
<b>Condition</b>	
Condition	None
ConditionParam...	
<b>Validation</b>	
Validation	None
ValidationPara...	

DataPath property points to input file.

Resizable Header

Select All On Focus

Pop-up menu lets you customize appearance of this property sheet.

Those are message specific properties. You can right click on property sheet to get popup menu and change ways how property sheet works.

List of properties:

Property	Description
Index	Each object in the map has unique index. This property is used internally by translator. It can also be useful if you decide to develop your own plug-ins for translator.
Name	Name of current selected map object.
TranslationType	Translation type is based on incoming message type. Translator uses specific processing logic for each message type.
DataPath	Data source or destination location. Must start with file:// or ftp:// or jdbc:// or other predefined source. Example of file data path: <a href="file://C:\test\ctrans\testfile.txt">file://C:\test\ctrans\testfile.txt</a>
SpecialInstruction	AppendToFile – translator will append all output data to one file in the output. This flag works on output side file mappings only. DeleteFileWhenDone – translator will delete input file, when processing for it is finished. This flag works on input side file mappings only.
ElementSep	Map element separator. It can be comma “,” for text files, or some EDI X12 element separator if your input file is X12.
SegmentSep	Separator for map segments. It can be carriage return and line feed for text files, or some EDI X12 segment separator if your input file is X12. You can enter carriage return and line feed using expression “#13#10”. Those are decimal codes for carriage return and line feed.
SubElementSep	Subelement separator. Used in X12, EDIFACT, HL7 translations. Most of a time it is “:” colon character.
SubElementSep2	Same as SubelementSep.
EscapeCharacter	Used in X12 and EDIFACT to allow characters used as segment or element separators to appear in data. Escape character is usually “?” question mark.
AutoDetect	If set to true, translator will try to detect ElementSep, SegmentSep, SubelementSep,

	SubelementSep2 properties.
Function, FunctionParameters	External function implemented in Java plug-in class. It is called when item is accessed during processing.
Script, ScriptParameters	BeanShell script. It can contain most basic Java language code. Read more about this feature in special chapter Script property.
Condition, ConditionType	Data is processed for this item and all items below if condition is met. Read more about this feature in special chapter Condition property.
Validation, ValidationType	Validation that should be performed on the map object.
Mandatory	Can be used on segment or element to indicate that item is mandatory and should always be present in the input or output.
Tag	Property has slightly different meaning based on what translation is used: for EDI X12 and EDIFACT it is used to identify segments, for XML it is used as XML tag, for text translations it is used as text data block indicator.
FixedLength	Used for fixed text translation only. Set to true if segment or element is fixed length. Can ONLY be used in flat file translations.
Length	Length of fixed text segment or element. It is used in fixed length text translation. Can ONLY be used in flat file translations.
Padding	Pad spaces on the left or right side. Can ONLY be used in flat file translations.
OnBegin, OnBeginParameters	External function is called when transaction begins.
OnCommit, OnCommitParameters	External function is called when transaction commits.
OnRollback, OnRollbackParameters	External function is called when transaction fails and has to rollback.
PrepareSQL	If set to true, translator will use prepared SQL statements that are generally faster than unprepared. Some older or not fully compliant JDBC drivers do not support prepared SQL statements. Set this property to false if JDBC driver is not capable to use prepared statements.
AutoCommit	If set to true will commit each SQL statement otherwise commit only after all operations with the database are finished.
SQL	Reserved for future use.
FieldType	Reserved for future use.

### DataPath property

It is data source or destination. DataPath can point to a file, FTP connection, database, etc. It has to start with file:// or ftp:// or jdbc:// or other predefined source.

Prefix	Description	Example
file://	Local or network file	file://C:\test\mytestfile.txt
ftp://	FTP file	ftp://user01:pass1234@ftp.foo.com/README.txt:type=i
http://	HTTP file	http://user01:pass1234@www.foo.com/README.htm
jdbc://	JDBC connection	jdbc://jdbc:odbc:TESTDATA;UID=user01;PWD=pass1234
stream://	Custom data reader and writer implemented in Java. Read more about it in Developers' Manual.	stream://mypackage.MyReaderClass

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.

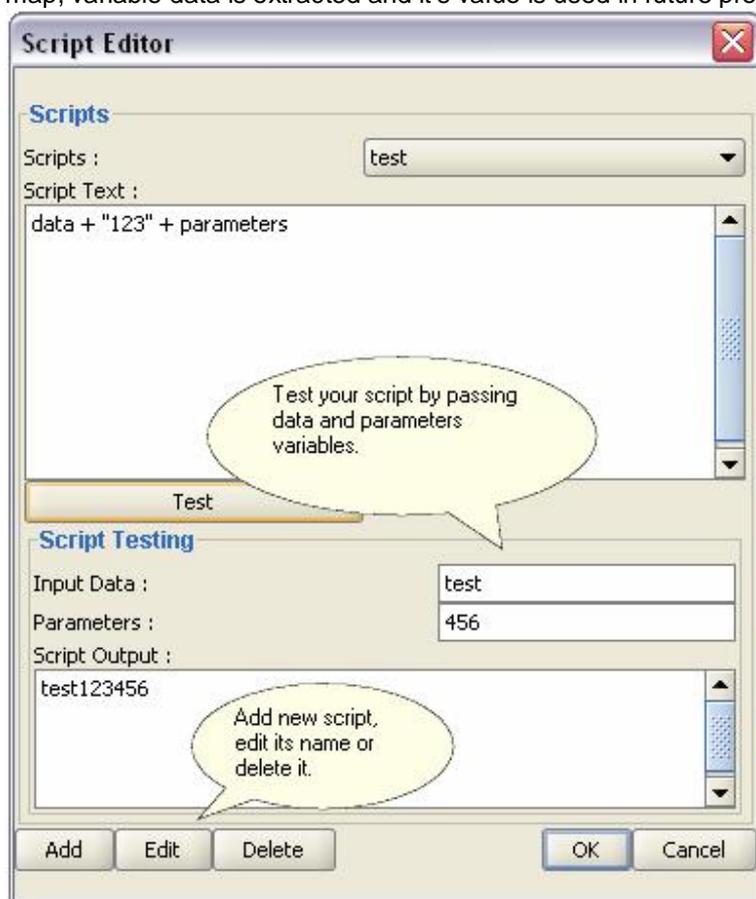
Macro	Usage
%Input%	Translator drops file extension from the input file, therefore any extension can be used in output DataPath property to form output file. Example: <a href="#">file://%Input%output.xml</a> - in this example output file

	name will be the same as input file but it will have "output.xml" appended to the end.
%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

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.

### Script property

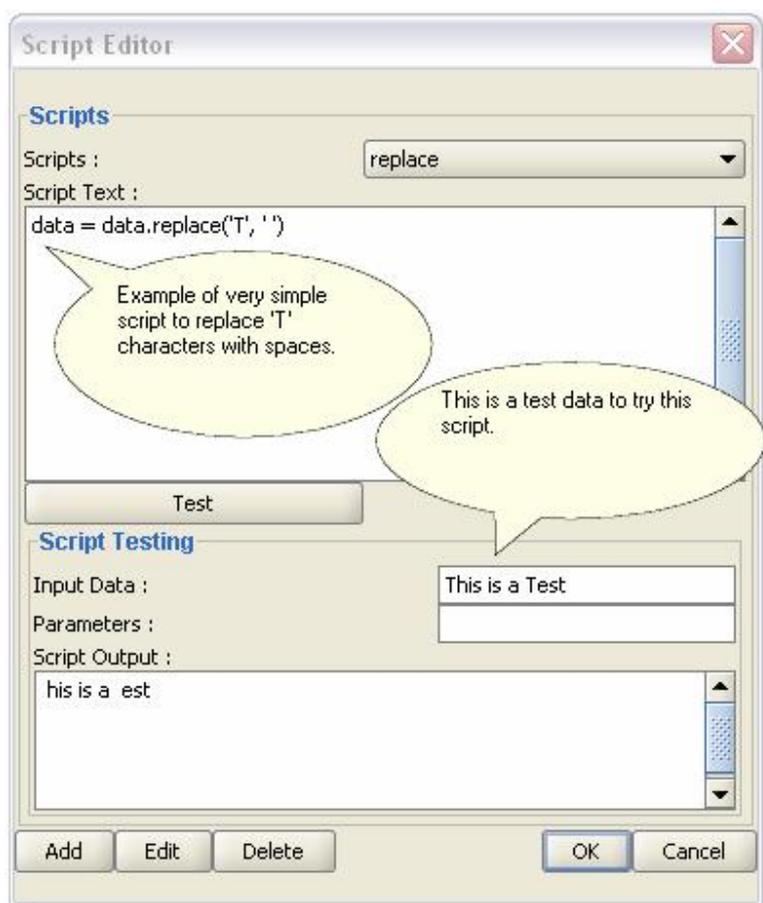
It uses BeanShell to execute your script in Java™. There are two variables passed into the script once it is executed: data and parameters. Variable called "data" is external data coming from data source defined via DataPath property. It can be entered for testing purposes in Script Editor but during actual map processing it comes from file or database. Variable "parameters" is what you can pass to script via ScriptParameters property. Once script is executed in the map, variable data is extracted and it's value is used in future processing.



Scripts can be added via scripts editor dialog.

Scripts are very powerful way to add custom functionality to your map without writing external plug-ins and classes in Java™. It is recommended to keep scripts short and under 20 lines. Longer scripts may become hard to manage, debug and test.

Scripts are excellent way to customize your map without writing custom plug-ins for it. However scripts are attached directly to the map and they are not external to it. They can become hard to manage if amount of developed maps grows beyond 20 and each map will have copies of the same set of scripts. In this case it could be better to move logic from scripts into external plug-ins.



Script to replace every instance of capital "T" in the input and change it with space.

### Function property

Function property lists plug-in functions that are external to translator engine, but can be called during processing to perform special formatting or access other data sources. Plug-ins can be coded in Java™ language and attached to almost every map object. Conditions and Validation properties are extendable via plug-ins as well.

```
/**
 * Trim spaces on both sides.
 */
public String trimSpaces(Integer id, String strData, String strParameters) {
    if (strData == null || strData.length() == 0) // return null if there is nothing left
        return null;
    return strData.trim();
}
```

Source code for external translator plug-in that trims spaces from incoming data.

### Condition property

Conditions are used to perform data processing branching. They are like "IF" statements in modern programming languages. Conditions can be used when certain values expected in the segment, element or field should be routed to specific output based on the incoming value.

Example: let say you have X12 segment N1 with specific first element qualifier. If this element value is NL then it is "ship to" address information and it has to go to "ship to" address fields on flat file output. You can create two N1 segments. First one with element 1 properties set Condition = NL and ConditionType = plugin.Check.isEqualTo would be mapped to "ship to" address on flat file and second N1 would have no conditions set and would take all other addresses.

Segment NM1 has condition set on first element to 40.

Conditions can be set to filter specific segments by qualifier values in incoming data.

Property	Value
<b>Common</b>	
Index	2125
Name	<b>Entity Identifier Code - 1</b>
Mandatory	<b>false</b>
<b>Processing</b>	
Tag	
FixedLength	<b>false</b>
Length	<b>0</b>
<b>Function</b>	
Function	<b>None</b>
FunctionParam...	
<b>Script</b>	
Script	<b>None</b>
ScriptParameters	
<b>Condition</b>	
Condition	<b>plugin.Check:isEqualTo</b>
ConditionParam...	<b>41</b>
<b>Validation</b>	
Validation	<b>None</b>

Row Number	Data for selected item	Length	Sequence
1	41	2	40

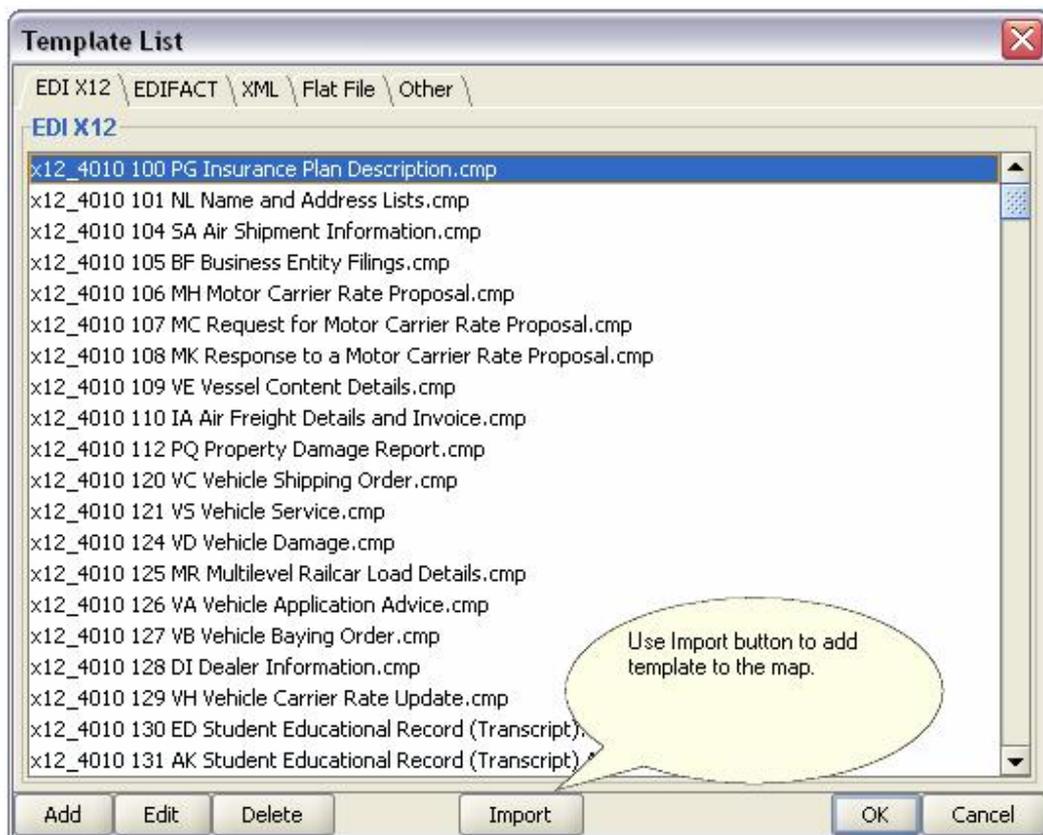
Completed processing

That is another example for conditions set on NM1 segment in X12 837. If first element is equal 41 then segment data should be retrieved.

#### Validation property

Validation and ValidationType properties are used to perform validation on data going through translator. Properties are provided to perform some basic validation on specific map object.





Templates can be added, edited in the list or deleted from the list using Add, Edit, Remove buttons. You can also use Import button to import (add) template to the current map.

Templates have the same file \*.cmp extension as map files. You can find more templates for other formats under translator installation directory.

## Translation

Translation is performed in this sequence:

1. Map is loaded.
2. Data is retrieved from input data source.
3. Specific data parser is activated based on input translation type.
4. Translator walks input/left side of the map.
5. Data parser breaks input stream of data in pieces and loads internal structures in the translator engine.
6. Specific data producer is activated based on output translation type.
7. Translator walks through output/right side of the map.
8. Data is collected and written to the output data source.

Every time translator hits segment or element that contains data following execution is performed in this order:

1. If there is external plug-in defined in Function property it is executed.
2. If there is script defined in Script property it is executed.
3. If there is condition set in Condition property it is evaluated. If segment or element does not meet condition set it is not processed.
4. If there is validation set in Validation property it is performed. If segment or element does not meet validation set error is reported.

Order of execution is very important as you may pre-process element's data using Function property and finish formatting using Script. Every time map object is processed order of execution can be illustrated as:

Function -> Script -> Condition -> Validation

What are the options this order of execution gives you? You can do a lot of pre-processing before Condition is evaluated and Validation is performed because they are called after Function and Script.

## EDI X12, EDIFACT translation

Easy way to start new EDI X12 or EDIFACT map is to load already existing template of specific X12 or EDIFACT message. You can do it via Template Wizard. Simply "Add" new template if it does not exist and use "Import" button to place it into the map. Template files have the same file extension as maps (\*.cmp).

After template is loaded you can change segment, element and subelement separators in properties SegmentSep, ElementSep and SubelementSep. If separators can not be predefined in the map because input data is coming from different trading partners and can have number of different separator values, then you can set to use AutoDetect property to "true" and hope that translator will find separators automatically. It is recommended to set separators if they are known and do not rely on AutoDetect property, data might be corrupted or separators might be slightly off and translator will fail to detect them. AutoDetect works by searching ISA segment in X12 or UNA segment in EDIFACT message. It does not search whole message only header. Once ISA or UNA is found, separators are extracted.

If your EDI X12 or EDIFACT message is non-standard you can still translate it but in that case map has to be created manually using "Add" menu. You would start with "Add Message", set message name, separators, DataPath property and then "Add Segment", "Add Element". Manual map creation usually takes 8-12 hours longer than using a template.

There is a table of properties for EDI X12 and EDIFACT. Most of properties have to be setup otherwise translation will not work properly.

Map Object	Property	Description
Message	DataPath	Property MUST point to actual input file or set of files. It can be FTP, HTTP location, local or network file or database connection string. DataPath has to start with prefix for the protocol. That can be file://, ftp://, http://, jdbc://.
	SegmentSep	You CAN set Segment separator for the message
	ElementSep	You CAN set Element separator for the message
	SubelementSep	You CAN set Subelement separator for the message
	AutoDetect	If it is set to "true" will try to detect input file or message separators. If it is set to "false", then properties SegmentSep, ElementSep and SubelementSep MUST be set.
Segment	Tag	It MUST be actual string used to find segment. Example: ISA, GS, ST, GE, etc.
Element	Mandatory	You SHOULD set it to "true" if element should be in the output but it is not mapped.
	Function, FunctionParameters, Script, ScriptParameters	You CAN use one of many build-in or custom plug-in functions for special formatting, counts, sequence numbers, etc.
	Condition, ConditionParameters	It CAN be set on element if segment processing is "Conditional", that is segment should be processed if one of it's elements contains some specific qualifier value. You can think of it as "IF" statement: "IF some element receives this constant value, then segment should be processed". This is very powerful concept, as you can add a few segments one after the other in the message, set elements in them to have different conditions and that way process segments with the same Tag but different element values in them. Read special chapter on "Condition property".
	Validation, ValidationParameters	It CAN be used on element to perform basic validation.
SubElement	Same as Element	

	except no Mandatory property	
--	------------------------------	--

If your input file is EDI X12 or EDIFACT and you have imported the template, then you can set DataPath property and run the map even if output (right) side is not defined. Translator will open file defined in DataPath properties, read and parse it. If there are any segments that cannot be processed they will be reported as warnings. After you ran the map, click on any item in the input map (element or segment) and click on "Data" tab, it will display data for that item.

If your output file is EDI X12 or EDIFACT then running the map will not produce any output, you will need to define at least some input message and create some mappings to get results.

## HIPAA support

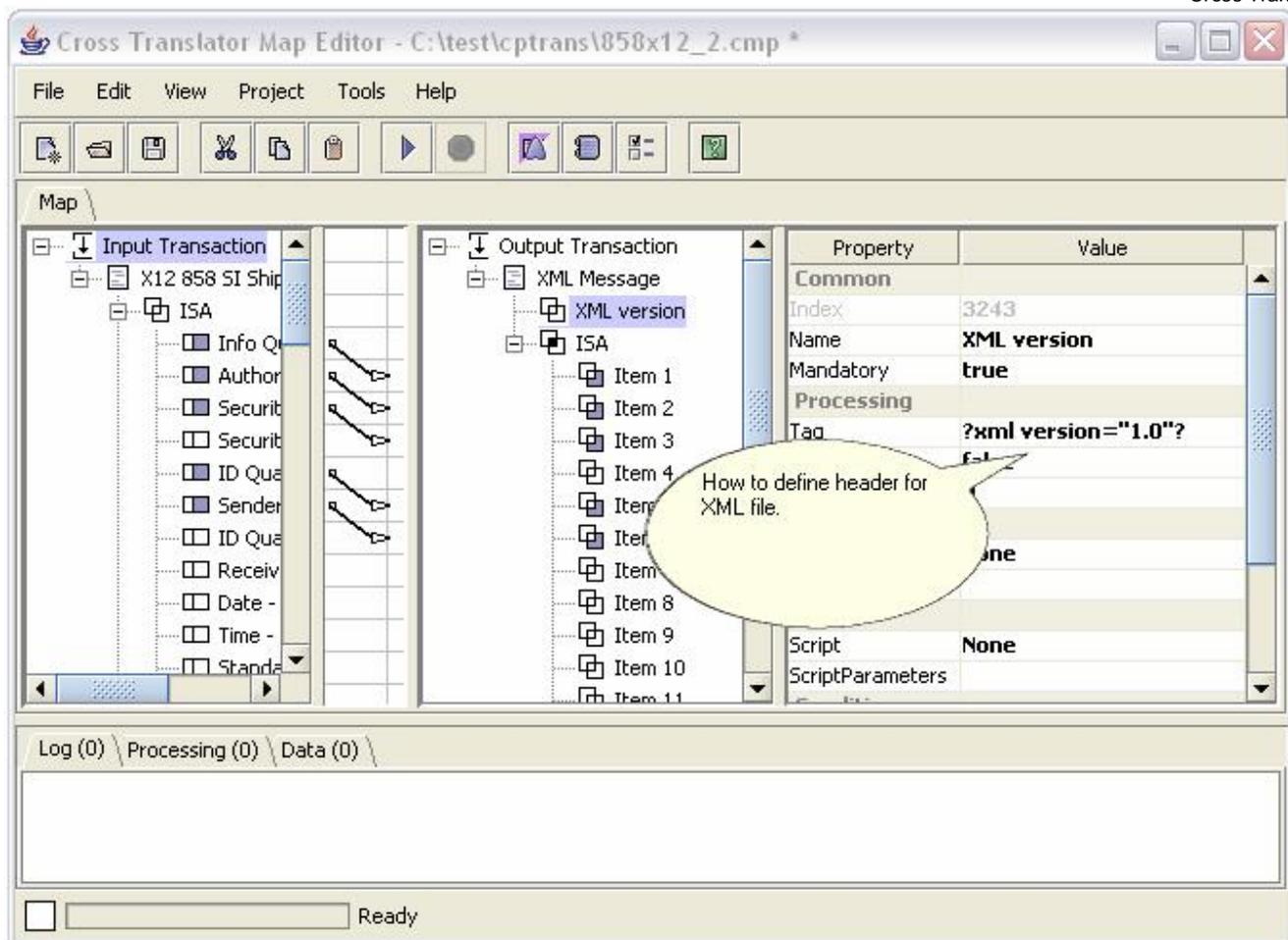
HIPAA Healthcare supported via HIPAA templates. You can choose one of the templates in "Template Wizard" in order to import it into your map. After you import it is the same mapping process as with typical EDI X12 file (see chapter on EDI X12 translation).

## XML translation

You can use some of the pre-built templates if source or result of your translation is EDI X12 and EDIFACT message. They are replicated copies of EDI X12 or EDIFACT message definitions. If no one of templates matches your requirements you can always create map manually by using "Add Message", "Add Segment", "Add Element" menus. Segment is actual XML tag that may contain other tags/segments. Element is XML tag attribute.

There is a table of properties for XML messages. Most of properties have to be setup otherwise translation will not work properly.

Map Object	Property	Description
Message	DataPath	Property MUST point to actual input file or set of files. It can be FTP, HTTP location, local or network file or database connection string. DataPath has to start with prefix for the protocol. That can be file://, ftp://, http://, jdbc://.
Segment	Tag	It MUST be actual XML tag string used to find or produce segment.
	Mandatory	You SHOULD set it to "true" if segment should be in the output but it is not mapped.
	Function, FunctionParameters Script, ScriptParameters	You CAN use one of many build-in or custom plug-in functions for special formatting, counts, sequence numbers, etc.
	Condition, ConditionParameters	It CAN be set on segment if segment processing is "Conditional", that is segment should be processed if one of it's elements or segments contains some specific qualifier value. You can think of it as "IF" statement: "IF some element receives this constant value, then segment should be processed". This is very powerful concept, as you can add a few segments one after the other in the message, set elements in them to have different conditions and that way process segments with the same Tag but different element values in them. Read special chapter on "Condition property".
	Validation, ValidationParameters	It CAN be used to perform basic validation.
Element	Same as Segment	



This is an example on how to define XML version tag at the beginning of XML file. Mandatory property should be set to true and Tag property should be set to ?xml version="1.0"?

## XML schema validation

Not implemented in version v1.x. Expected in later releases. It will be implemented via plug-in plugins.XML.validateSchema. Plug-in function is already defined, however validation code is not implemented yet.

## Flat file translation

### Flat text file with delimiters

Most common and used flat text file format is Comma Separated Value (CSV) format. It is a single line of text ending with carriage return and line feed. Each element (field) in it is separated by comma. You can start mapping by adding Text message to the transaction, and adding one segment to that message.

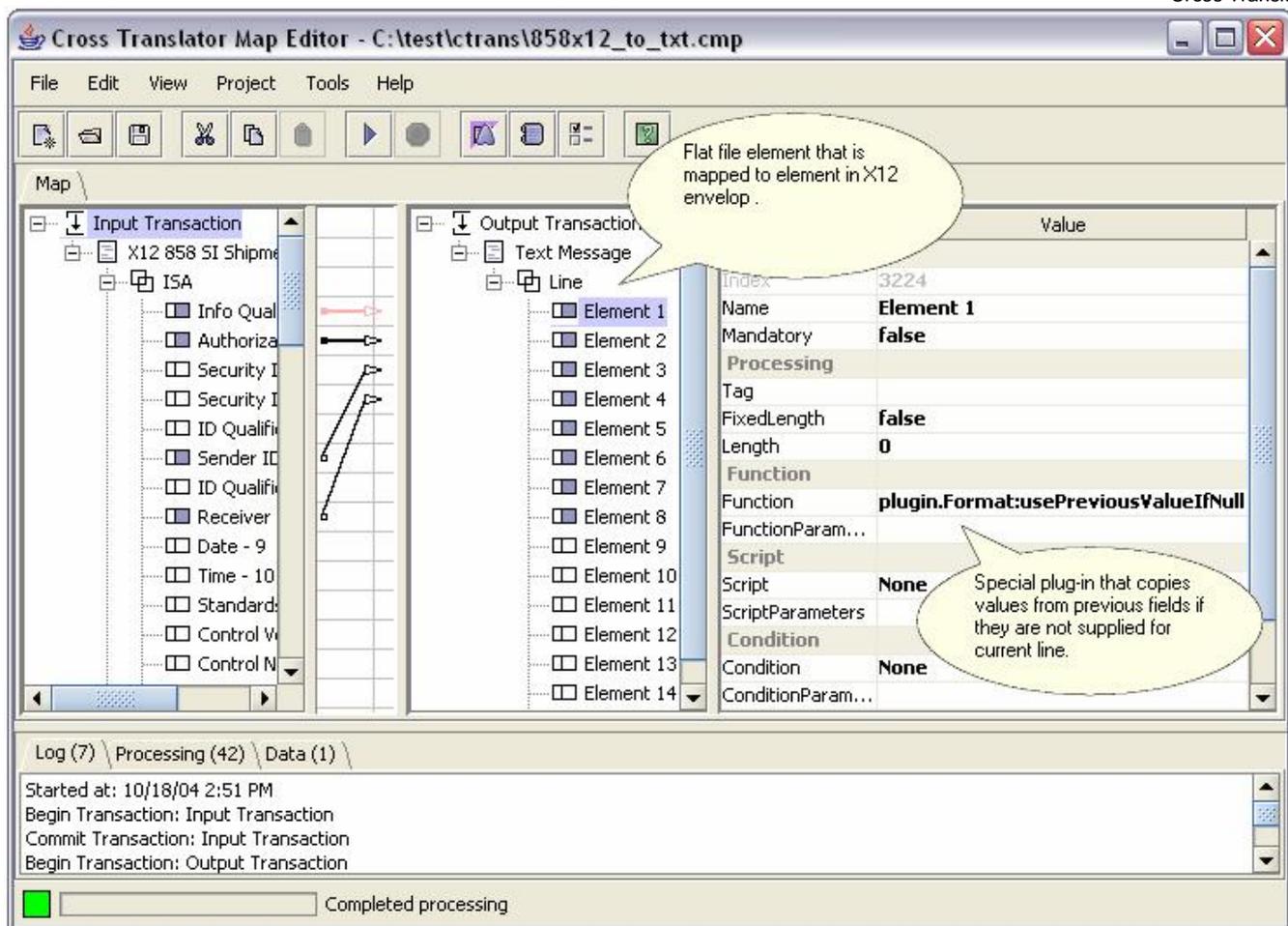
There are the steps:

1. Add message using right click pop-up menu on the transaction.
2. Fill DataPath property with [file:///C:/some\\_directory/myfile.txt](file:///C:/some_directory/myfile.txt).
3. Set SegmentSep to #13#10 (decimal code for carriage return and line feed CRLF).
4. Set ElementSep to comma or other character that will be used as text field separator.
5. Add segment to the message.
6. You can set Tag property for this segment if you want each line to be prefixed with some constant value.
7. Add elements to the segment.
8. Each element is a field in text file. If it is output text file and element is not mapped then it should be set to Mandatory = true.
9. You can also set other properties for the element like Function, Script, etc.

There is a table of properties for flat file with delimiters messages. Most of properties have to be setup otherwise translation will not work properly.

Map Object	Property	Description
Message	DataPath	Property MUST point to actual input file or set of files. It can be FTP, HTTP location, local or network file or database connection string. DataPath has to start with prefix for the protocol. That can be file://, ftp://, http://, jdbc://.
Segment	Tag, FixedLength, Length	It CAN be actual string of characters used to find or produce segment. If Tag is not used then FixedLength MUST be set to true and Length MUST be set to actual segment length in characters.
	Mandatory	You SHOULD set it to "true" if segment should be in the output but it is not mapped.
	Function, FunctionParameters Script, ScriptParameters	You CAN use one of many build-in or custom plug-in functions for special formatting, counts, sequence numbers, etc.
	Condition, ConditionParameters	It CAN be set on segment if segment processing is "Conditional", that is segment should be processed if one of it's elements or segments contains some specific qualifier value. You can think of it as "IF" statement: "IF some element receives this constant value, then segment should be processed". This is very powerful concept, as you can add a few segments one after the other in the message, set elements in them to have different conditions and that way process segments with the same Tag but different element values in them. Read special chapter on "Condition property".
	Validation, ValidationParameters	It CAN be used to perform basic validation.
Element	Same as Segment	

If you are producing text file from looping data like XML, EDI X12 or EDIFACT some elements might come from the header that loops only once and some elements of text file might come from the detail that loops somewhere from 1 to 10000 times. Translator will try to line up and output loops flat however that means that header data will be empty for some details. If you want each header line in flat file to repeat per each unique detail line you can use special plug-in on flat file elements that come from header. It is Function property called plugin.Format.usePreviousValueIfNull.



Special function used to repeat header data for detail lines when mapping from looping data source like XML, EDI X12 or EDIFACT to flat files.

### Flat text file fixed length

Please read previous chapter “Flat text file with delimiters” as they share some common features.

Fixed length text record files have fields that are fixed in length and padded with spaces or zeros on the left or right. It is a single line of text ending with carriage return and line feed. You can start mapping by adding Text message to the transaction, and adding one segment to that message.

There are the steps:

1. Add message using right click pop-up menu on the transaction.
2. Fill DataPath property with [file:///C:/some\\_directory/myfile.txt](file:///C:/some_directory/myfile.txt).
3. Set SegmentSep to #13#10 (decimal code for carriage return and line feed CRLF).
4. Add segment to the message.
5. You can set Tag property for this segment if you want each line to be prefixed with some constant value.
6. Add elements to the segment.
7. Set element's FixedLength property to true and Length property to actual field length.
8. Each element is a field in text file. If it is output text file and element is not mapped then it should be set to Mandatory = true.
9. You can also set other properties for the element like Function, Script, etc. Some of them can be used to pad element with spaces or zeros. Check Appendix 1 of this document for complete list of plug-ins that can be used in Function property.

There is a table of properties for flat file fixed length messages. Most of properties have to be setup otherwise translation will not work properly.

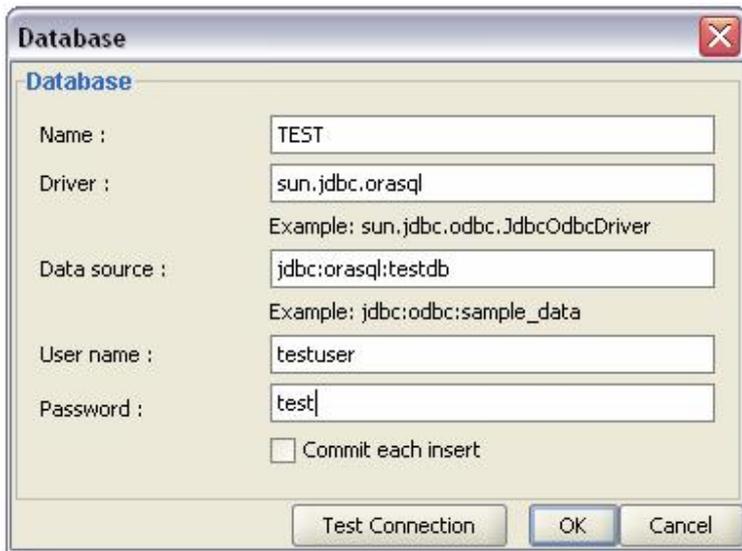
Map Object	Property	Description
Message	DataPath	Property MUST point to actual input file or set of files. It can be FTP, HTTP location, local or network file or database connection string. DataPath has to start with prefix for the protocol. That can be file://, ftp://, http://, jdbc://.
Segment	FixedLength, Length, Padding, Tag	FixedLength MUST be set to true and Length MUST be set to actual segment length in characters. If FixedLength is set to false then Tag property should be string of characters used to find or produce segment.
	Mandatory	You SHOULD set it to "true" if segment should be in the output but it is not mapped.
	Function, FunctionParameters, Script, ScriptParameters	You CAN use one of many build-in or custom plug-in functions for special formatting, counts, sequence numbers, etc.
	Condition, ConditionParameters	It CAN be set on segment if segment processing is "Conditional", that is segment should be processed if one of it's elements or segments contains some specific qualifier value. You can think of it as "IF" statement: "IF some element receives this constant value, then segment should be processed". This is very powerful concept, as you can add a few segments one after the other in the message, set elements in them to have different conditions and that way process segments with the same Tag but different element values in them. Read special chapter on "Condition property".
	Validation, ValidationParameters	It CAN be used to perform basic validation.
Element	Same as Segment	

## Database mappings

### Basics

Database mappings are supported in Cross Translator. Current version supports mapping to and from database tables via JDBC connection. Relationships are supported via master-detail queries on the input side (data comes from SQL queries) and Primary-Foreign Key relationships for output side when records are being inserted into the database. In essence each table is treated as separate flat data structure, in a way similar to how flat text files with delimiters are processed.

Your JDBC driver has to be JDBC compliant and it has to be in the CLASSPATH for translator to load and use it. If you use translator engine packaged in JARs, it is easy, simply add classpath to your JDBC driver to the CLASSPATH passed into translator JARs. However if you use translator engine or tools packaged in EXE on Windows OS then you can set classpath using special variable CT\_PLUGINS in your system environment to point to JDBC driver classes or JARs. Classpath in CT\_PLUGINS is appended and passed to translator engine once Windows executable starts.



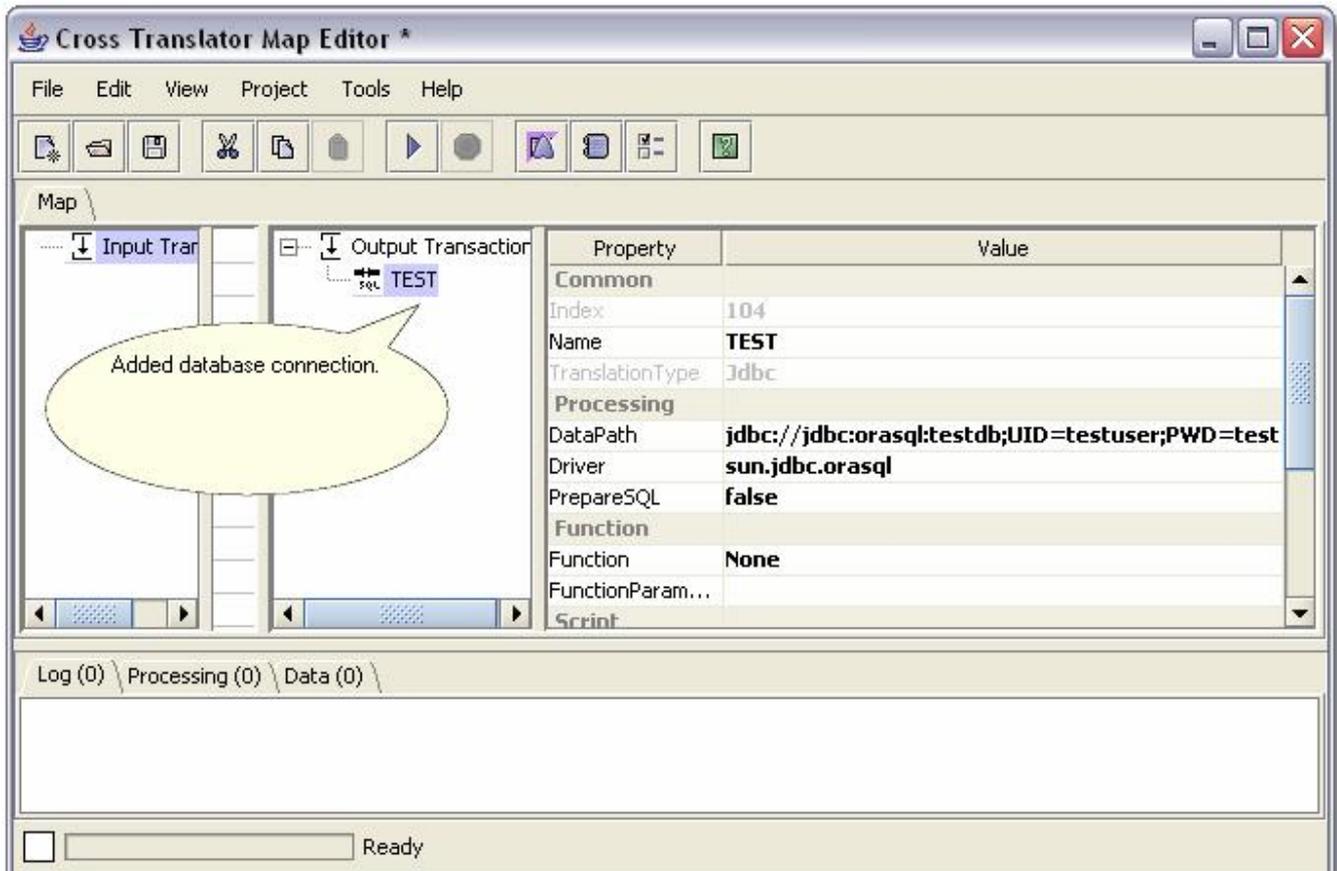
Add database connection using "Add Database" menu.

JDBC drivers are available for most of relational databases. They are usually packaged in JAR files that come with relational database client software installation package. Please consult your relational database vendor for JDBC driver availability.

Special note:

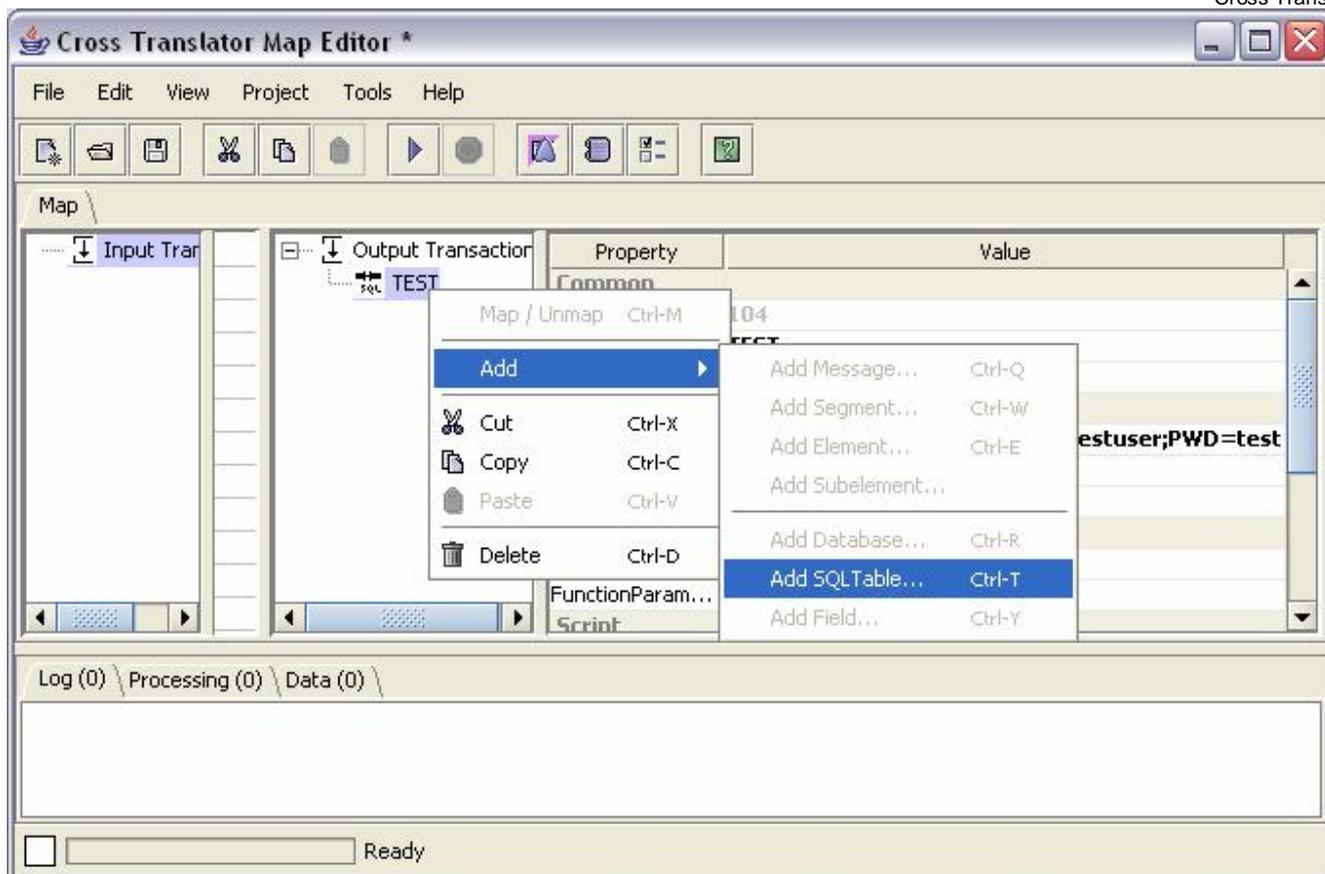
mySQL JDBC driver is very sensitive on parameters it receives. You must provide database name in DataPath property otherwise your SQL statements will be failing. Example of DataPath that worked in our tests:

`jdbc://jdbc:mysql://localhost:3306/Test?user=monty&password=greatsqldb`  
 "Test" is our database name. Use your database name instead.



This is database connection added to the map.

PrepareSQL property should be set to false for JDBC drivers that do not support prepared SQL statements.  
AutoCommit property should be set to true if each transaction to the database should be committed automatically.



This is example of adding SQL Table to the database connection in the map.

### Master detail queries

Master-detail relationships can be setup in the map. Translator will execute master query and for each row of the result set, it will call detail query with parameters formed from the values of master fields. Master-detail queries are good choice if you have tables containing header and detail information, and you want to produce results in ordered fashion with each header row grouped with detail rows. Example: single invoice header information containing invoice line items.

Master query can be referenced in detail query SQL by using following syntax

:MASTER\_QUERY.MASTER\_FIELD

If Detail query field is a string of characters, then it should be placed in single quotes

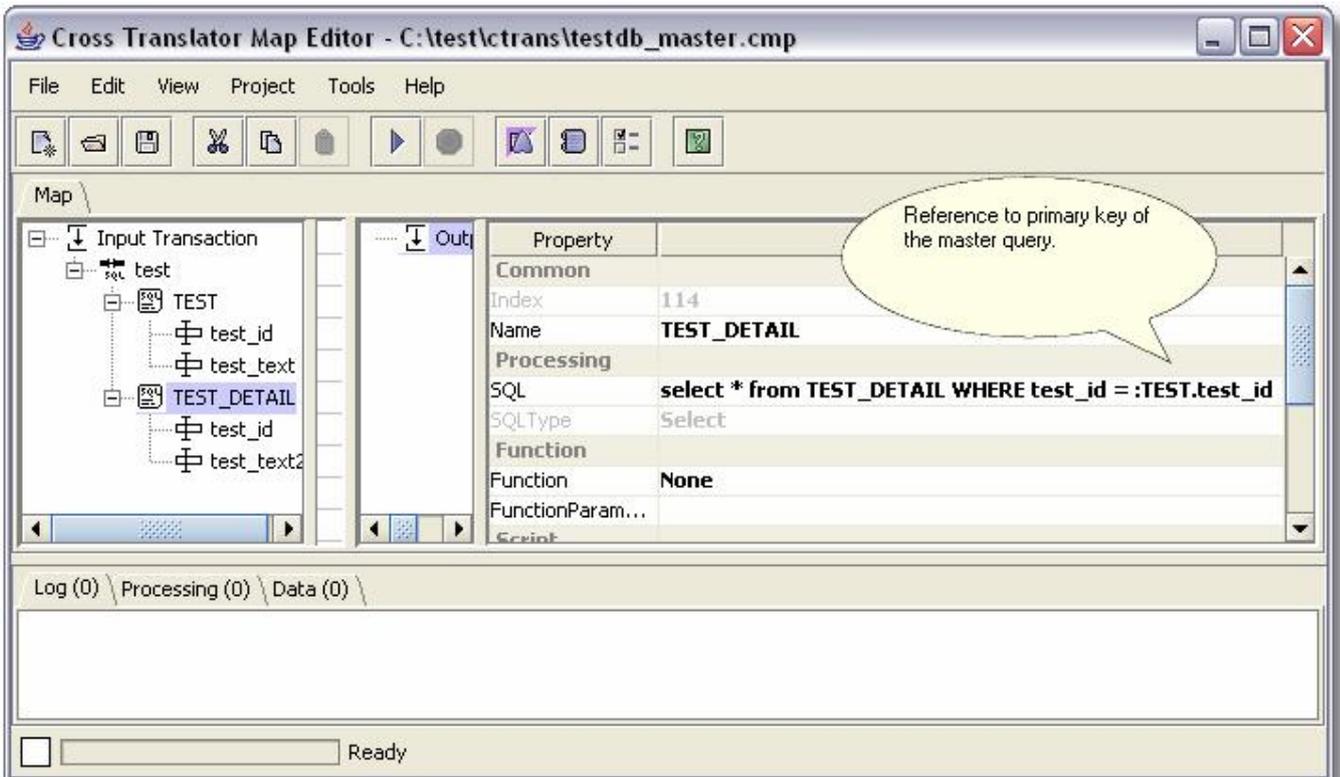
':MASTER\_QUERY.MASTER\_FIELD'



If field is a string of characters place it is single quotes, otherwise database types will not match.

### Important restrictions:

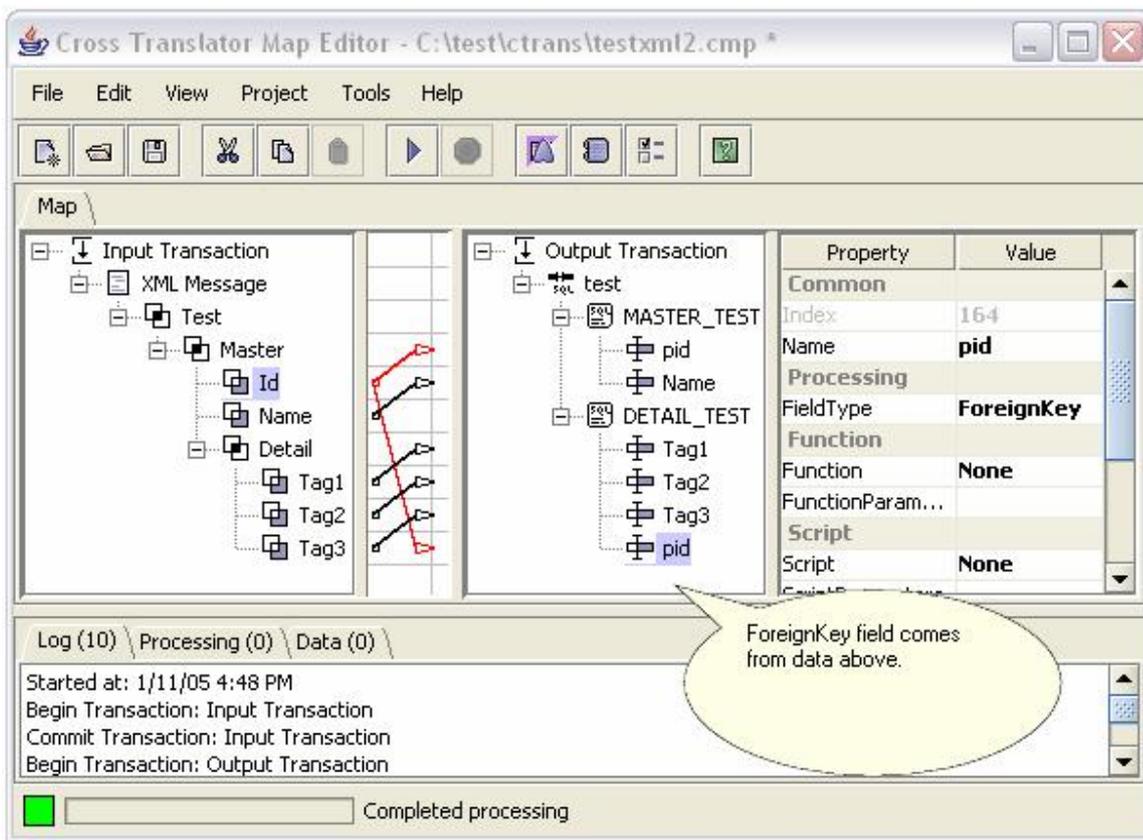
1. One detail query cannot reference more than one master query.
2. Two detail queries cannot reference each other. Example: if query A would reference and use some master fields from query B, and query B would use master fields from query A then it would not be clear which query should run first.



SQL statement may reference other query by its map name and field. Master query is named TEST.

#### PrimaryKey and ForeignKey fields

It is recommended to mark one field in the mapped table to be of FieldType = PrimaryKey. This field will be used to line up data values coming to all other fields. Translator can load data into database tables with relationships. Tables can have FieldType set to PrimaryKey or ForeignKey based on the type of relationship they are involved in the database.

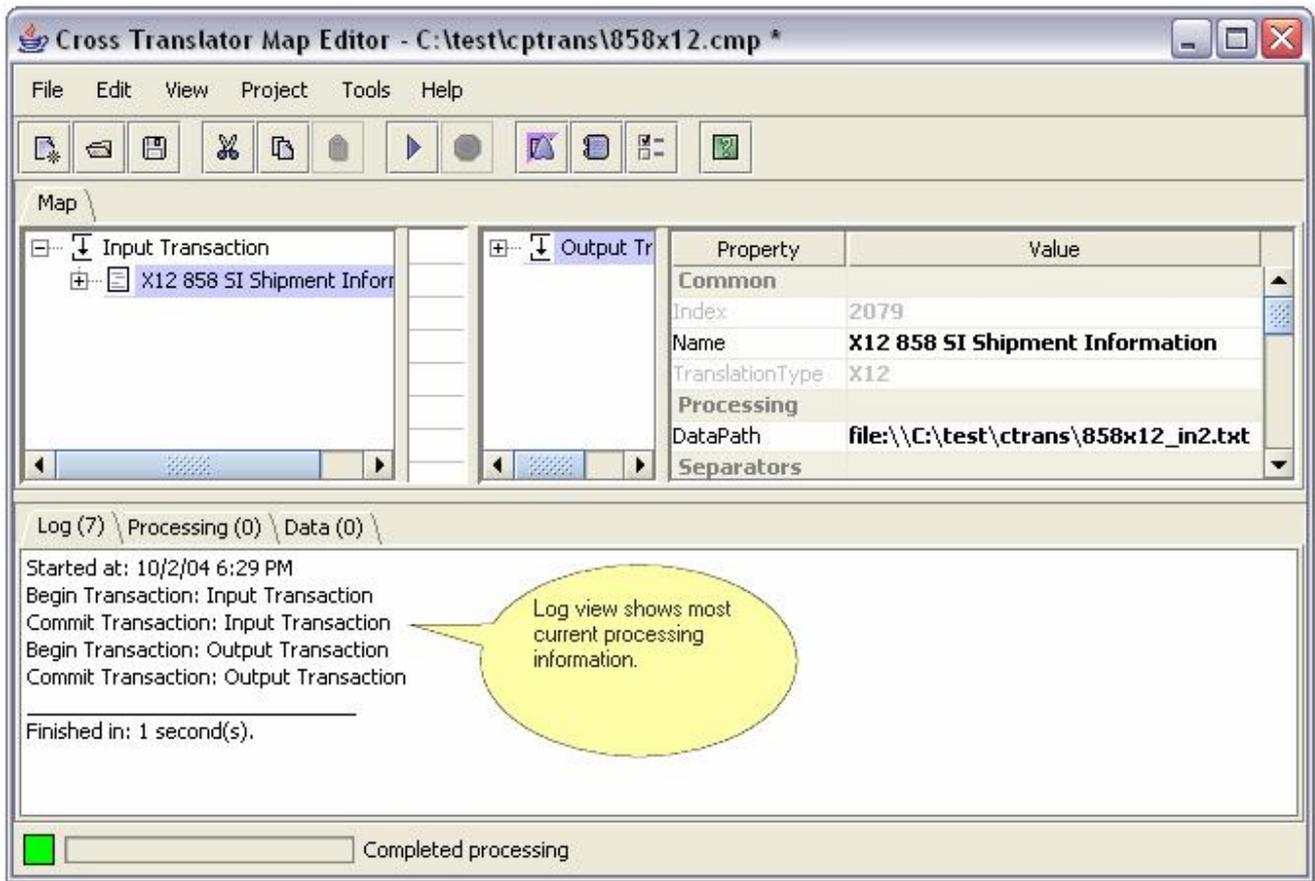


ForeignKey is used on fields that are related to the data in other tables.

Each table should have one field that has FieldType=PrimaryKey. If non of the fields has FieldType=PrimaryKey on the table translator will produce warning during processing. Other possible field types are Nullable and NotNull. NotNull is used in cases when you only get a few field values inserted in the bunch of records and want to fill the blanks with values from the previous records. Once FieldType is set to NotNull on the field translator will try to reuse value from the previous record if current record does not have a value.

## Log tab

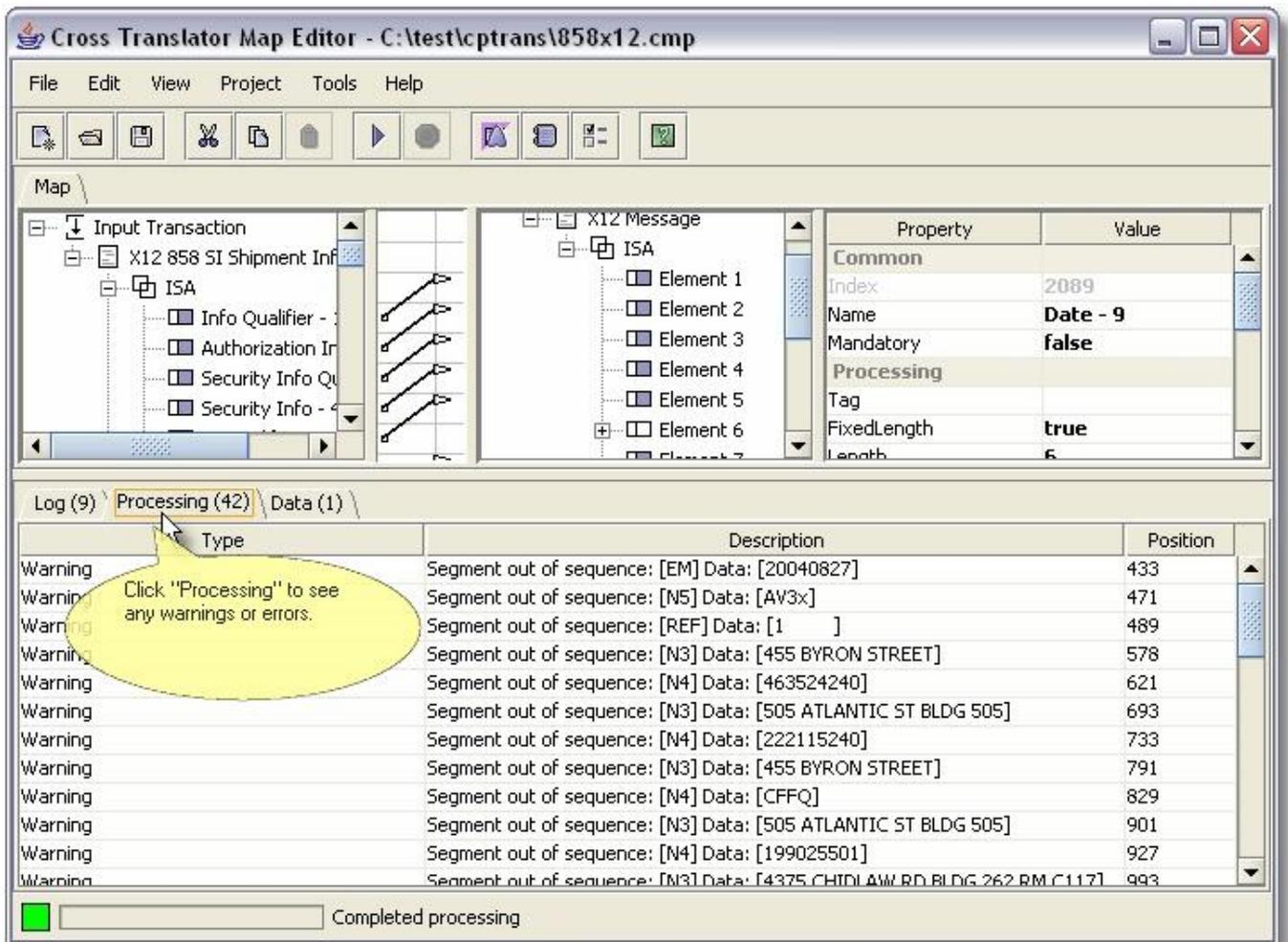
It provides messages from translator to the map. Log may contain error messages and even printed execution stack trace if translation fails. All the information provided in Log tab also goes to log file. Log file contains most of information about current processing and all processing's since the last time log file was created or recycled. If processing fails for no clear reason and you cannot resolve the issue, you should contact technical support and email your log file. Log file can be defined in Options. If it is not defined, it will default to map file name with \*.log prefix. For example if your map is my\_translate.cmp, then your log file will be my\_translate.cmp.log.



This is typical log view when processing is successful.

## Processing tab

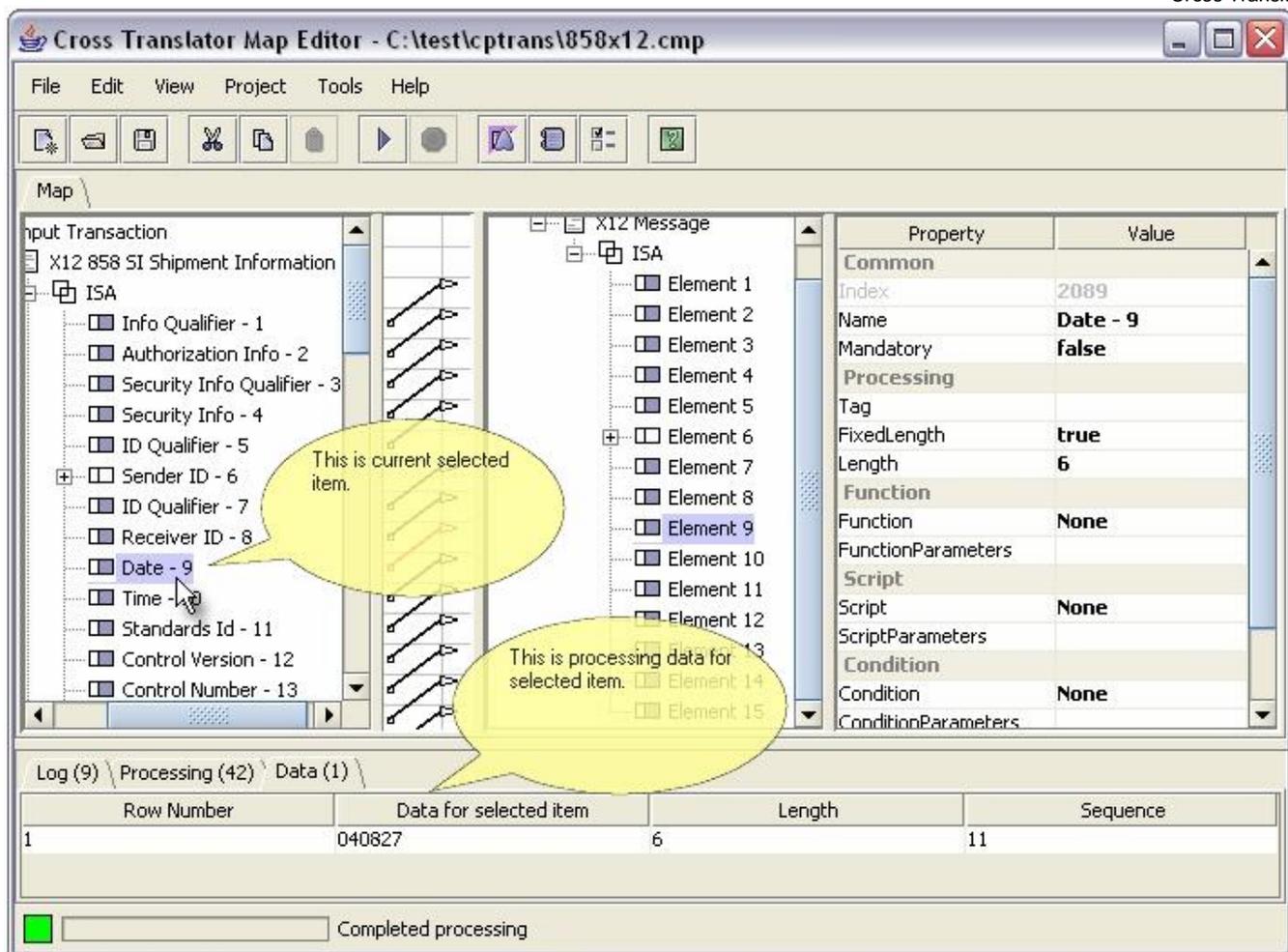
Processing tab shows any non-critical warnings, information messages or errors occurred during processing of the map. Warnings and errors in this list are not vital to the processing and translator was able to recover and continue. However it is best to get rid of all the warnings and errors in this list before map is used in production environment. You can limit number of rows displayed in the list via Options screen setting.



This is list of warnings and errors.

## Data tab

You have to run the map to see data in this tab. Data tab displays processing data for selected item in the map. If item is looping or has many occurrences there will be many rows in the list. Sequence column in the table is internal translator number that is used to keep track how input data items came in and how output should be laid out. Data tab may not contain any rows if output is produced via plug-in or script.



Data list for map item currently selected.

## Performance Tuning

Performance can be tuned for better user experience or faster processing. There are command line parameters that can be passed into any translator utility including Map Editor to make it run faster or use computer resources more efficient. All the options discussed below may not work for specific Java™ Runtime (JRE) and may even prevent utilities to run. It is better first try not to use any specific options and just try to run utilities using default or no parameters.

All Java™ -X type parameters can be passed to translator utilities even if utility is packaged into EXE. However if utility is used under Windows™ and is used as EXE application any JRE options have to be prefixed with -J option.

For example: you can set initial size of memory for translator to use (so called "heap") if you run Map Runner with parameters -J-Xms[size].

In this example ctranrun.exe runs with 64Mbt initial heap size:

```
>ctranrun.exe -J-Xms64M
```

Also you can use -J-Xmx[size]

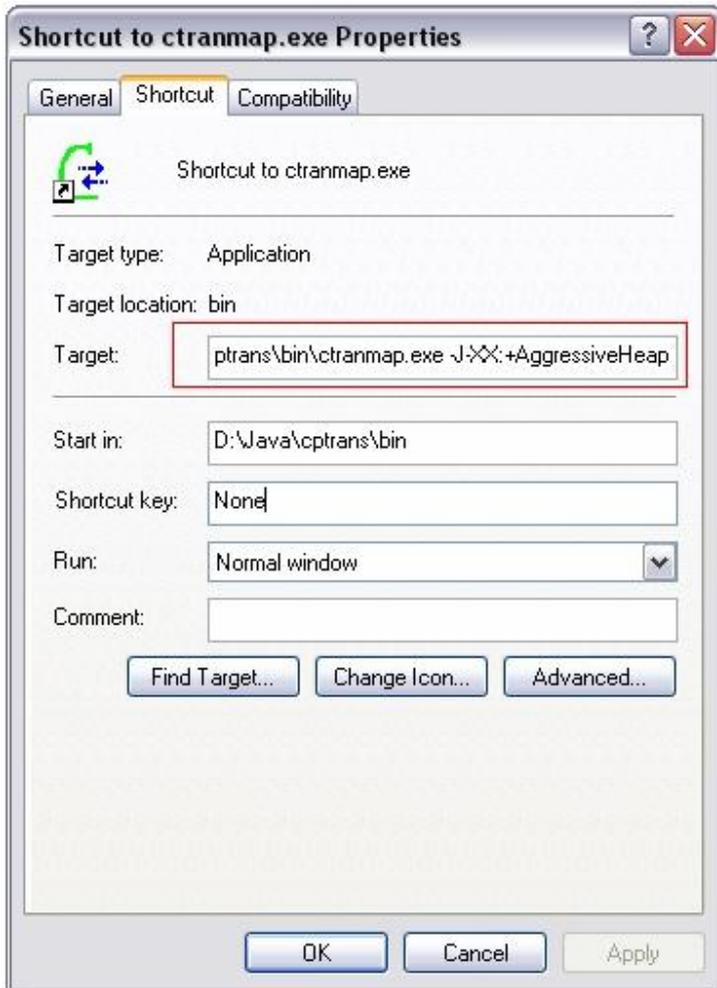
In this example ctranmap.exe runs with 256Mbt maximum heap size:

```
>ctranmap.exe -J-Xmx256M
```

Be careful with maximum heap size parameter, if it is set too low translator utility will fail with OutOfMemory error.

If you have standalone dedicated server that will be running maps and processing data constantly, it might be better to give it all computer's memory and resources to run at full speed. It can be done using -XX:+AggressiveHeap

option. This is an example on how to run map using translator command line utility on a server:  
 >ctrancmd.exe -J-XX:+AggressiveHeap [other command line options]



Aggressive heap option means that utility will use as much memory as needed to process the map as it would be running on the computer alone. It is not recommended if you need to run some other applications on the machine at the same time.

There are special options we recommend for Map Editor if you are using JRE 1.4.2:  
 -Xcongcg -XX:MaxPermSize=128m

Full command line would look like:  
 >ctranmap.exe -J-Xcongcg -J-XX:MaxPermSize=128m

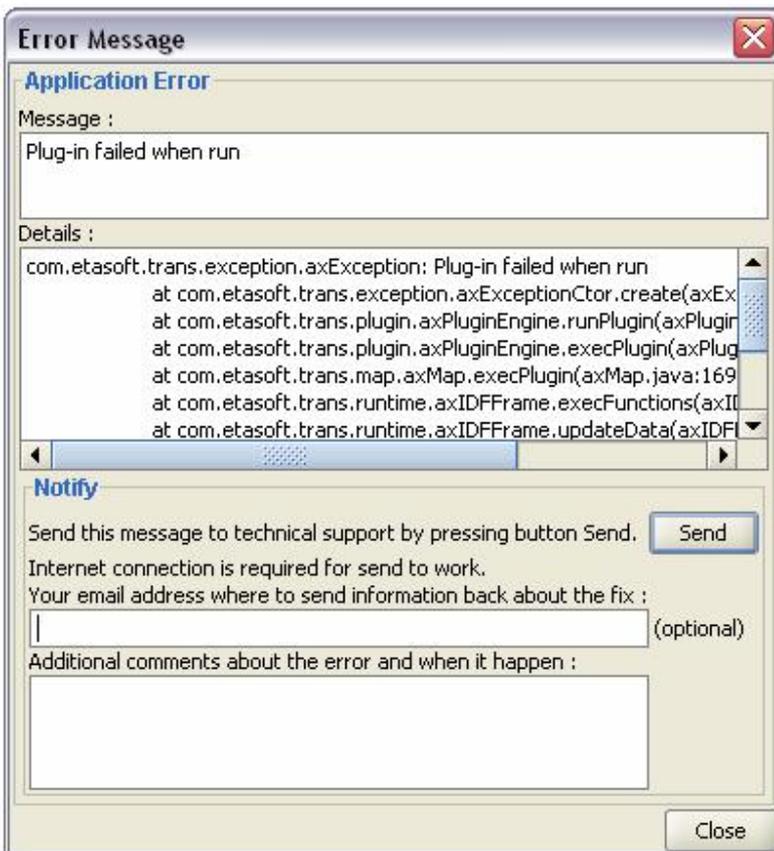
That will turn on concurrent garbage collector and should minimize pauses when you are creating maps. However in total time concurrent garbage collector will run longer.

## Technical Support

Contact information for technical support is published on the product's web site. There is an option to send error information directly from the application. When error message pops up click on "Show More Details" button, then fill your email address and additional comments in Error Message screen and click "Send". Your email address is optional but if it is not supplied we will not be able to respond to you about the fix.

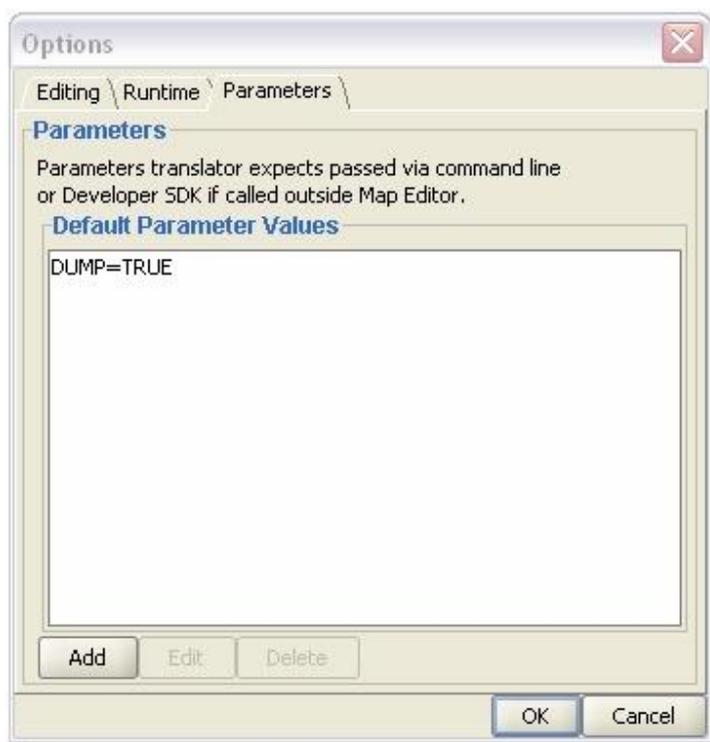


Screen above shows how application errors are reported.



Screen to report errors to technical support. This feature will work only if Internet connection is available.

There is also special parameter that can be used to enable extra logging abilities of the translator. Extra logging can be used to diagnose looping and other serious mapping issues. If `DUMP=true` parameter is passed to translator internal data presentation will be produced in the log. Text in log window can be selected if you click on it and press `CTRL+A` on the keyboard. Once selected it can be copied with `CTRL+C` and pasted with `CTRL+V`. Log window results are duplicated in the log file.



This is special parameter to enable additional information in the log.

## Appendix A. List of build-in plug-ins

This is a list of all plug-ins that come with the default installation. Functions can be used in Function, Condition, Validation, OnBegin, OnCommit and OnRollback properties. Some functions are used for data manipulation and formatting, and some are functions for conditional processing, meaning if result is true then condition is met and processing continues.

One example of conditional function is "isEqualTo". Function checks if input value matches value in Parameters property. If it does, then condition is met and segment is processed, if it does not then segment is ignored. If this function is used in Validation then in case if condition is not met, error is reported.

Example of data manipulation function is "getDefaultValue". It simply takes value from Parameters and places in to the output. It can be used in cases when some default values should be inserted into the output data.

List of built-in plug-ins:

Plug-in function	Parameters property	Usage
All functions below are used for data manipulation in <b>Function</b> property.		
plugin.Format.getDefaultValue	String	Takes value from Parameters and places in the output. Can be used to place constant values in the output data. If there is no Parameters defined throws exception.
plugin.Format.getDefaultValueNoParam	String	It is similar to plugin.Format.getDefaultValue. Takes value from Parameters and places in the output. Can be used to place constant values in the output data. If there is no Parameters defined returns incoming data.
plugin.Format.getDefaultValueIfEmpty	String	Same as getDefaultValue, but places constant into the output only if input value is empty or null.
plugin.Format.getDefaultValueIfNotEmpty	String	Opposite of getDefaultValueIfEmpty.
plugin.Format.getEmptyIfNull	None	Converts NULL values into empty string. Can be useful in cases if nothing is produced in the output, not even tags or empty EDI segments, because all incoming data is NULL. In most cases it should be used on the input side.

plugin.Format.trimSpaces	None	Trims spaces from left and right side of data
plugin.Format.trimLeft	Character	Trims specific character from the left
plugin.Format.trimRight	Character	Trims specific character from the right
plugin.Format.usePreviousValueIfNull	None	Places previous value from the loop or line above if current value for the map item does not exist or is empty.
plugin.Format.padSpacesLeft	Length to pad to	Pad data with spaces on the left based on number entered in Parameters property.
plugin.Format.padSpacesRight	Length to pad to	Pad data with spaces on the right based on number entered in Parameters property.
plugin.Format.padZerosLeft	Length to pad to	Pad data with zeros on the left based on number entered in Parameters property.
plugin.Format.padZerosRight	Length to pad to	Pad data with zeros on the right based on number entered in Parameters property.
plugin.Format.removeCharacter	Character	Removes specific character from the data
plugin.Format.reformat	Pattern String	This is a special data reformatting function especially designed to reformat dates, short data blocks of predefined length. Parameters property should contain reformat pattern with escape characters @ and _. Read about reformat plug-in at the end of this table.
plugin.Format.substitute	Pairs of input=output separated by comma	Function designed to substitute certain values of input into specific values for the output. FunctionParameters property should contain string in a format: inputvalue1=outputvalue1,input2=output2,defaultvalue
plugin.Format.substituteFromFile	File name where pairs of input=output are stored	Works the same as plugin.Format.substitute but takes pairs of input1=output1,input2=output2 values from external file. Values should be separated by comma. Example of parameters: C:\directory\mysubstitutes.txt
plugin.Format.combineToStart	Parameter identifies group name to combine	Function designed to combine input data elements based on the Parameter value. Should be used on the input side to append various blocks of input file into one final element mapped to the output side. Output: combinedData + currentData.
plugin.Format.combineToEnd	Parameter identifies group name to combine	Function designed to combine input data elements based on the Parameter value. Should be used on the input side to append various blocks of input file into one final element mapped to the output side. Output: currentData + combinedData.
plugin.Format.combineReset	Parameter identifies group name for combined data	This function should be used with functions plugin.Format.combineToStart or plugin.Format.combineToEnd. It resets combined data cache to initial value.
plugin.Format.combineResetAll	None	This function should be used with functions plugin.Format.combineToStart or plugin.Format.combineToEnd. It resets combined data cache to initial values for all combined data.
plugin.Count.get	Count Name	Returns current value for the count
plugin.Count.getIncrement	Count Name	Increments current value for the count and returns it
plugin.Count.increment	Count Name	Increments current count value
plugin.Count.getDecrement	Count Name	Decrements current value for the count and returns it
plugin.Count.decrement	Count Name	Decrements current count value
plugin.Count.reset	Count Name	Resets count to 0
plugin.Count.resetAll	None	Resets all counts to 0
plugin.Sequence.getSequence	File name with full path	Returns sequential numeric values and saves them in the file. It can be used when generated values should be unique per each processing. Example: EDI X12 or EDIFACT control numbers. Example of parameters:

		C:\directory\mysequences.txt
plugin.Transaction.startSegmentCount	None	Start segment count from this point on. Each segment will be counted and count kept until plugin.Transaction.getSegmentCount or plugin.Transaction.getSegmentCountMinusOne is called.
plugin.Transaction.getSegmentCount	None	Return segment count. Can be used in SE01 for EDI X12 or UNT01 for EDIFACT.
plugin.Transaction.getSegmentCountPlusOne	None	Same as plugin.Transaction.getSegmentCount but produces number greater by 1 in the output.
plugin.Transaction.getSegmentCountMinusOne	None	Same as plugin.Transaction.getSegmentCount but produces number less by 1 in the output.
plugin.Transaction.getSegmentCountMinusTwo	None	Same as plugin.Transaction.getSegmentCount but produces number less by 2 in the output.
plugin.EDI.currentDateYYMMDD	None	Current date. Example: 050131
plugin.EDI.currentDateCCYYMMDD	None	Current date. Example: 20050131
plugin.EDI.currentTimeHHMM	None	Current time in 24 hour clock. Example: 1815
plugin.checkFilter	Value not to be filtered (allowed value)	This function works together with function plugin.isFilter. It should be used above plugin.isFilter to check for the specific value. If that specific value is present in the input element or segment then filtering is turned ON. After it is turned ON subsequent use of plugin.isFilter will filter all incoming values.
plugin.setFilterTrue	None	This function works with function plugin.isFilter and plugin.checkFilter. It is used to reset the value of the filter to True and turn filtering ON.
plugin.isFilter	None	This function works with function plugin.isFilter and plugin.checkFilter. It filters incoming data when filtering is turned ON.
All functions below are used for conditional processing and/or validation in properties <b>Condition, Validation.</b>		
plugin.Check.isEqualTo	String	Checks if incoming value equals to string of characters in Parameters.
plugin.Check.isEqualToIgnoreCase	String	Same as isEqualTo but ignores case.
plugin.Check.isStartWith	String	Checks if incoming value starts with string of characters in Parameters.
plugin.Check.isEndWith	String	Checks if incoming value ends with string of characters in Parameters.
plugin.Check.isNumeric	None	Checks if incoming value is numeric.
plugin.Check.isInteger	None	Checks if incoming value is integer number.
plugin.XML.validateSchema	XML file name to validate	Not implemented yet.
All functions below are used for conditional processing and/or validation in properties <b>OnBegin, OnCommit, OnRollback.</b>		
plugin.FileAction.deleteFiles	File path	File path may end with wildcard pattern. Example: C:\test\ctrans\*.txt will delete all text files in the directory C:\test\ctrans\
plugin.FileAction.backupFiles	File path	File path may end with wildcard pattern. Example: C:\test\ctrans\*.txt will rename all text files in the directory C:\test\ctrans\ by adding "~" to the extension.

**plugin.Format.reformat** plug-in examples:

Simple example would be if you have year and month coming in a form "2004/10" and you want output to be "10/2004" then your Parameters property should be @5@6/@1@2@3@4. That means: take character at position 5 and place into the output, take character at position 6 and place into the output, etc.

@ with the number - means position in the input data.

Another example of use would be in case if incoming data should be surrounded by some constant value in the output. Your incoming data is year in a form of two characters "04" and you need to prefix it with "20" to make it "2004". Your Parameters property should be 20\_. This means that all incoming data should be placed where "\_" underscore is.