Etasoft
Building blocks of integration

**XTranslator guide on complex flat text file to EDI X12 translation**

Main website http://www.etasoft.com
XTranslator website http://www.xtranslator.com

**Basic requirements**

This document describes the process of mapping flat comma separated text file to EDI X12 message for translation.

In order to do the mapping you should have:

1. Original flat file with delimited fields. Fields can be delimited with comma or other delimiter.
2. Documentation explaining EDI X12 message layout and structure. You should have some document that would list all the EDI X12 segments and elements that you need to produce from the flat text file.

Contact your trading partner or supplier for simple documentation on EDI X12 message you need to produce from the flat file. This documentation usually lists some specific required segments and provides EDI X12 message number and release version number. In this document we have chosen to translate flat file to EDI X12 837 release 4010. Translator comes with number of templates provided via Template Wizard.

Once mapping is done you do not have to recreate it again simply save it into the file with extension *.xmp. You can run map files using other utilities that come in the package (read User's Manual about other utility programs).

Sample input file contains data with fictitious names and address data.

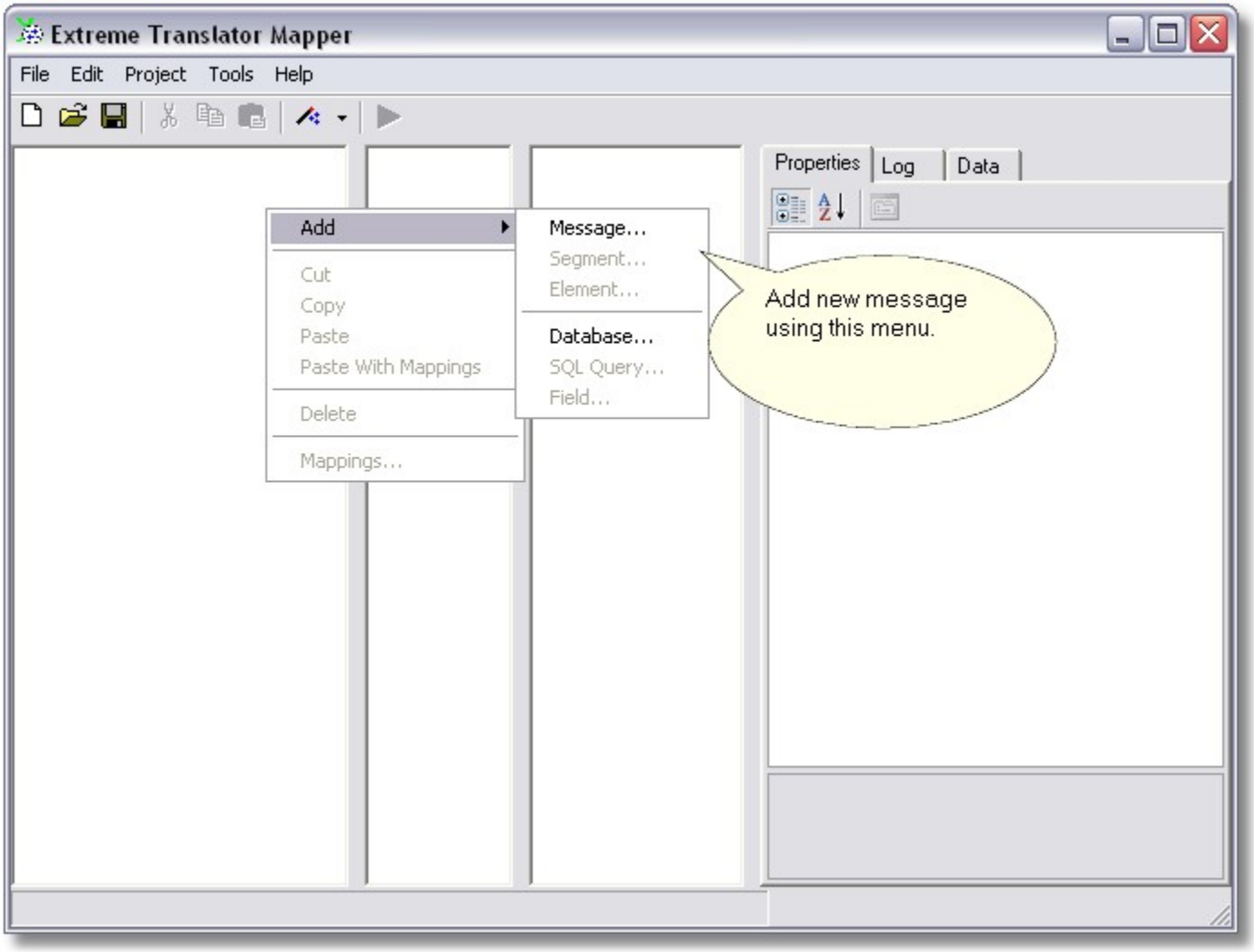
This document comes with complete map in a file with extension xmp. Please open and examine it as you read this document.

**How to define text file layout**

You can import flat file layout using By-Example Wizard. This Wizard simply reads all delimited fields and creates mapping objects for them. Wizard works on single repeating rows of CSV file.

Problem is that EDI X12 837 format has deep nested structure with multiple levels of HL segments. It is very difficult to produce 837 from single repeating rows of CSV file. We have to create multi line CSV file with header and detail lines. By-Example Wizard cannot create such layouts.

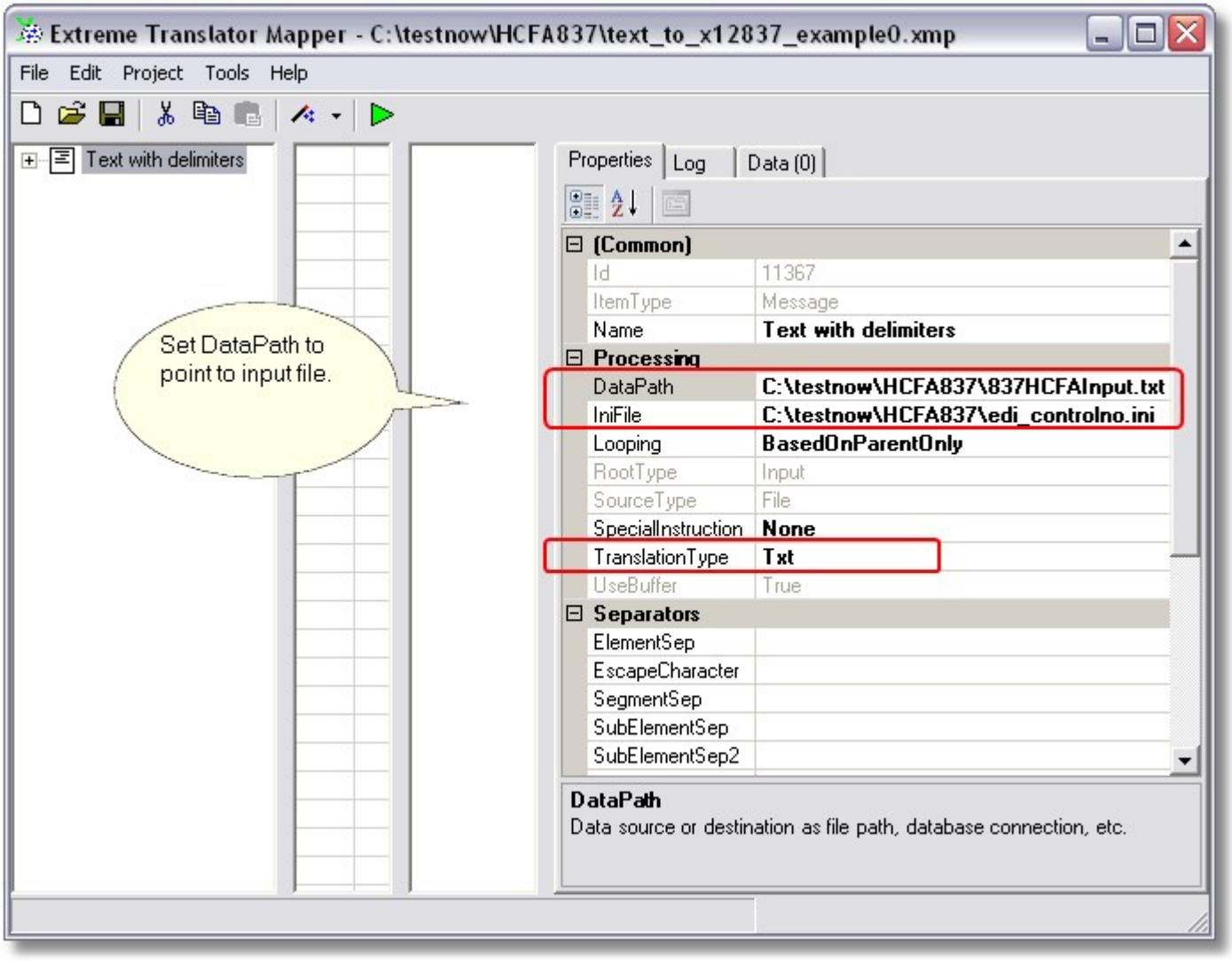
We will define it manually.




We add message using Add->Message pop up menu.

Manual setup takes longer than wizard as each input field has to be added using Add menu.


Choose “Manual mapping” and “Translation Type” Text.



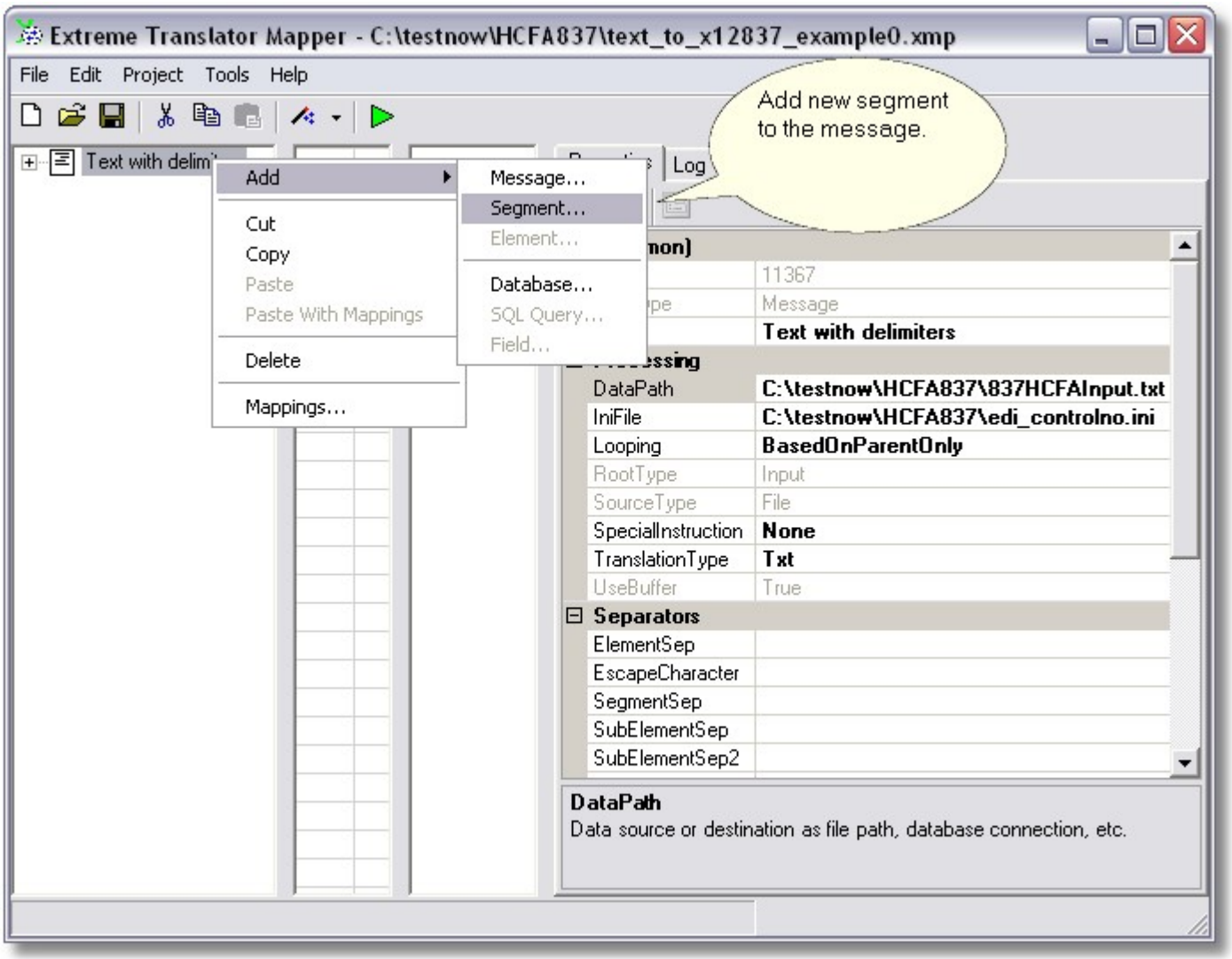

New item will be added to the map. This new item represents incoming file.

We need to setup few properties in order to read incoming file.

DataPath should point to incoming file. DataPath allows use of wildcards such as * so translator would pickup number of files. Example: DataPath setup to C:\test*.txt would pick up all the text files in C:\test directory.

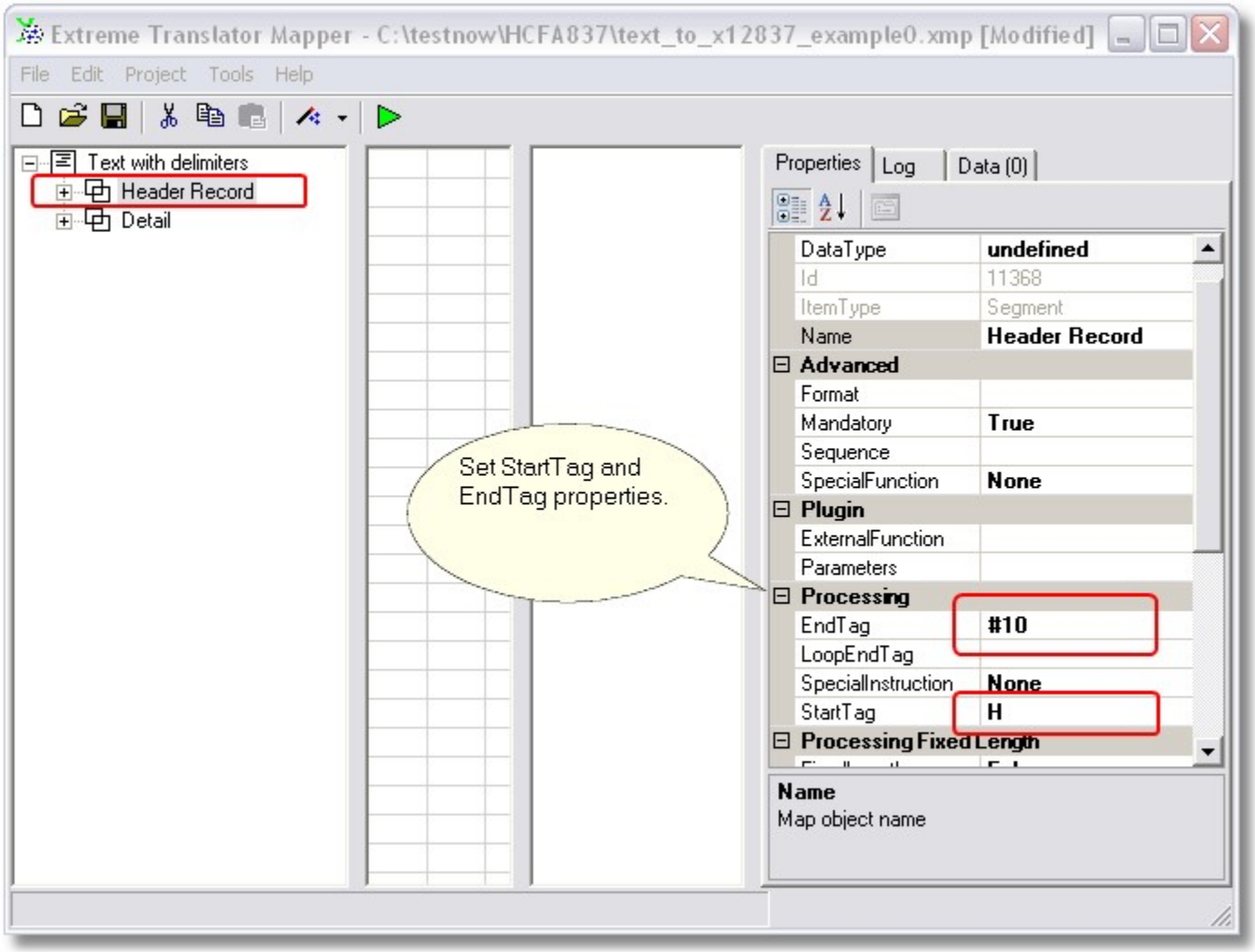
Please check User's Manual documentation on other options available in DataPath property.

You can also set IniFile property. This property works in tandem with property Sequence. It is used to generate and store control and other sequential numbers. There is Sequence property usage example below in this document.




Flat file usually has carriage return and line feed terminated lines or other block or line end terminators.

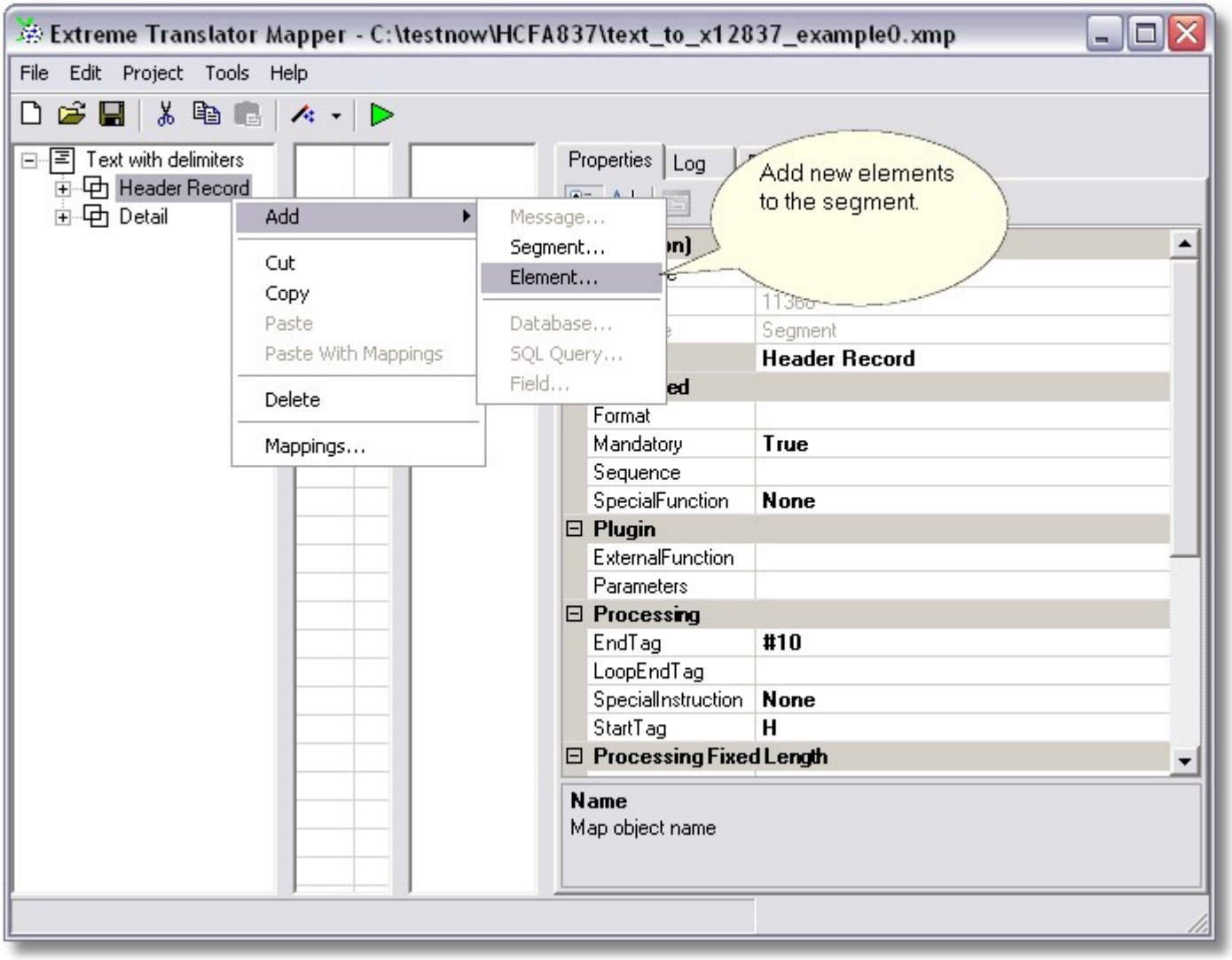
Segment represents block of text or line. Once you define segment you can add elements to it. Each element is a flat file field.



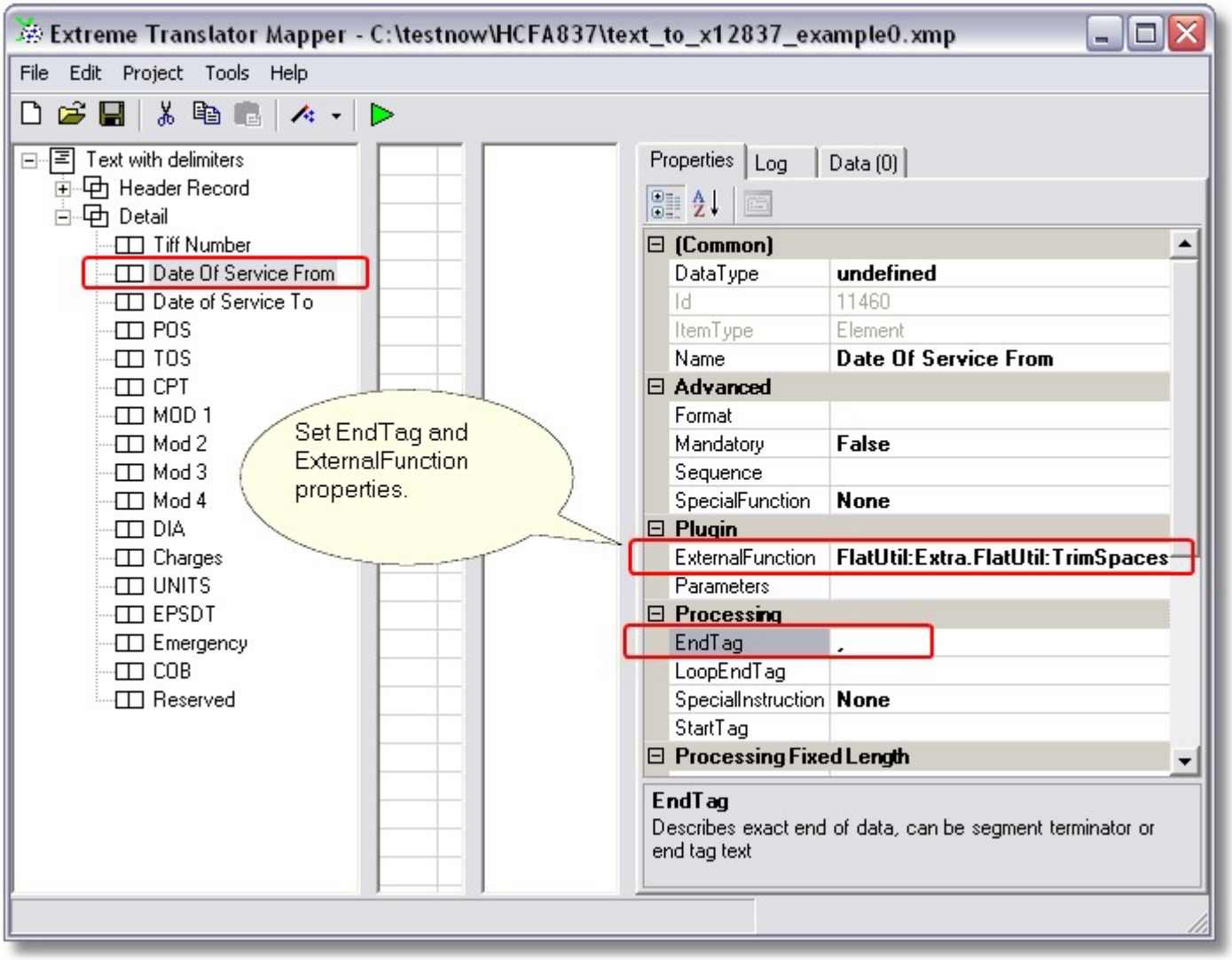

We add a segment and set its StartTag and EndTag properties.

StartTag property should hold incoming text that indicates start of the segment. EndTag property should hold segment end text or data. In the above picture StartTag is set to character “H” which means that once translator finds “H” in the input it will assume it is “Header Record” start. EndTag is set to #10 (decimal code for line feed character). Once translator finds “H” and locates line feed it will know that this data is the segment we are interested in. In most cases you can also set EndTag to #13#10 which is carriage return and line feed.

Our input file has header and detail lines therefore we have defined two segments each one with specific StartTag value (“H” for header and “D” for detail).




Each element represents single field in the flat file. We can add elements using Add menu.




EndTag property is used to mark end of the flat file field.

In our example each field is terminated by comma. There are number of properties you can use to format and transform incoming data. In this example we used ExternalFunction property to trim spaces.

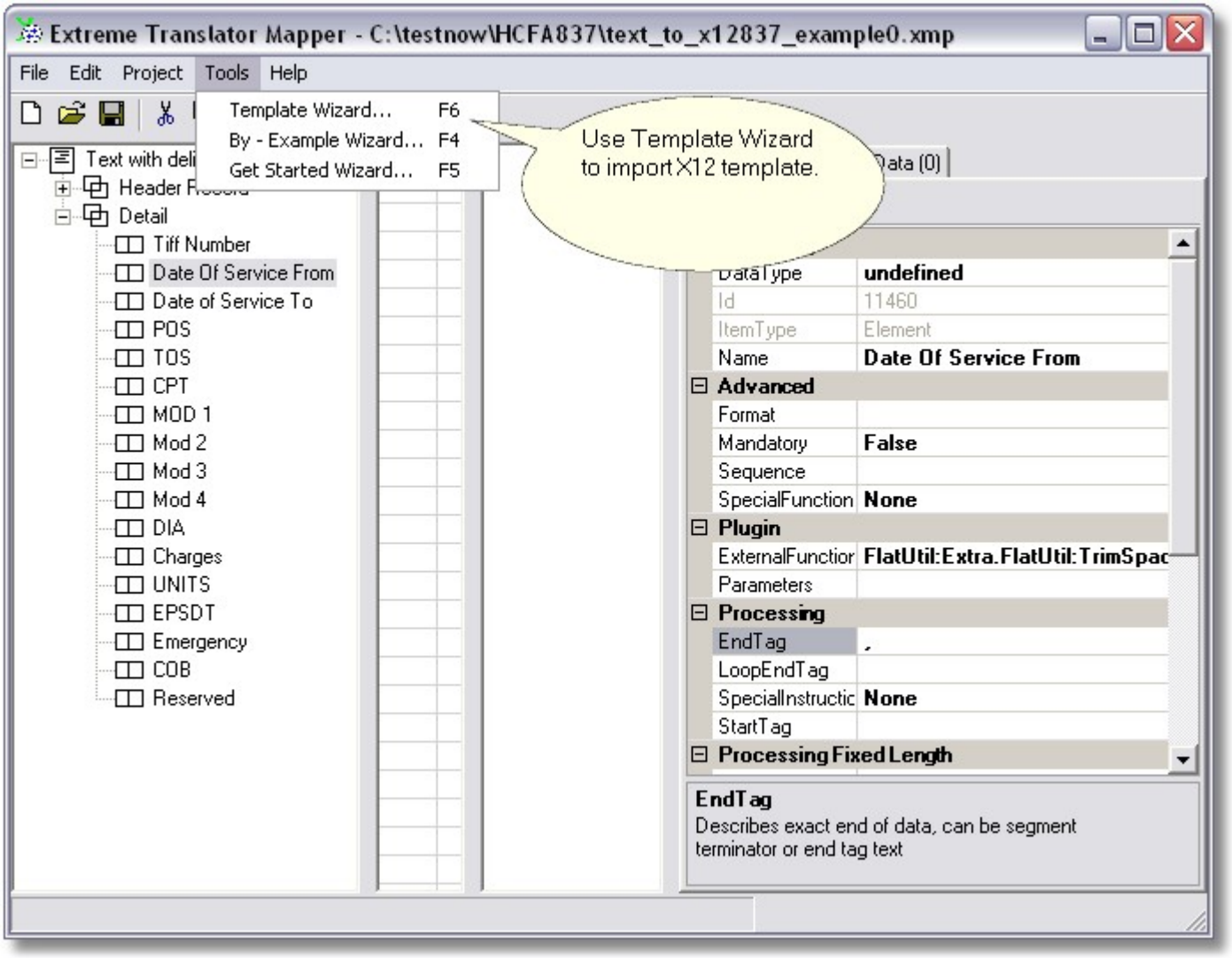
**How to define EDI X12 layout**

You can import EDI X12 layout using template from Template Wizard. Many EDI X12 templates are included with the product. We update and add more templates with every new release of the product. All templates are stored in \templates directory. Templates are grouped by they release version.

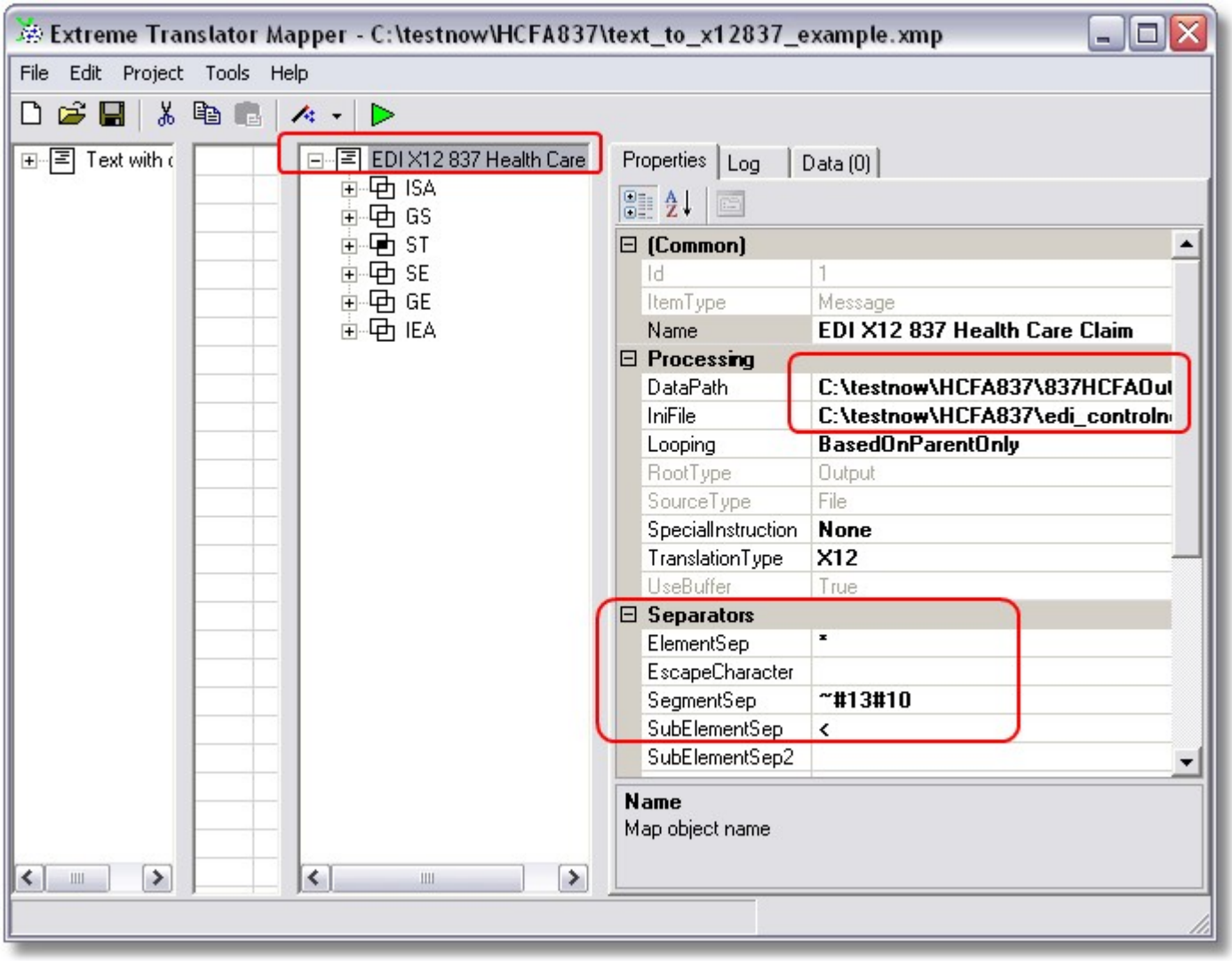
Translator includes templates for major standard releases. It may not have all the minor releases.

It is important to understand the difference between major and minor standard releases. In EDI X12 major releases numbered with schema “xx10” having last two numbers as “10” always. Translator has major releases 4010, 5010, 6010, etc. as templates but has no minor releases such as 5040 or 6030.
Most of a time only major releases are used for EDI implementations. If your trading partner requires minor release you can simply use major release as substitute. Example: use 5010 instead of 5040.

Some very small changes to the template might be required in order to adjust template for minor release.




Use Template Wizard to add specific EDI X12 template to the map.

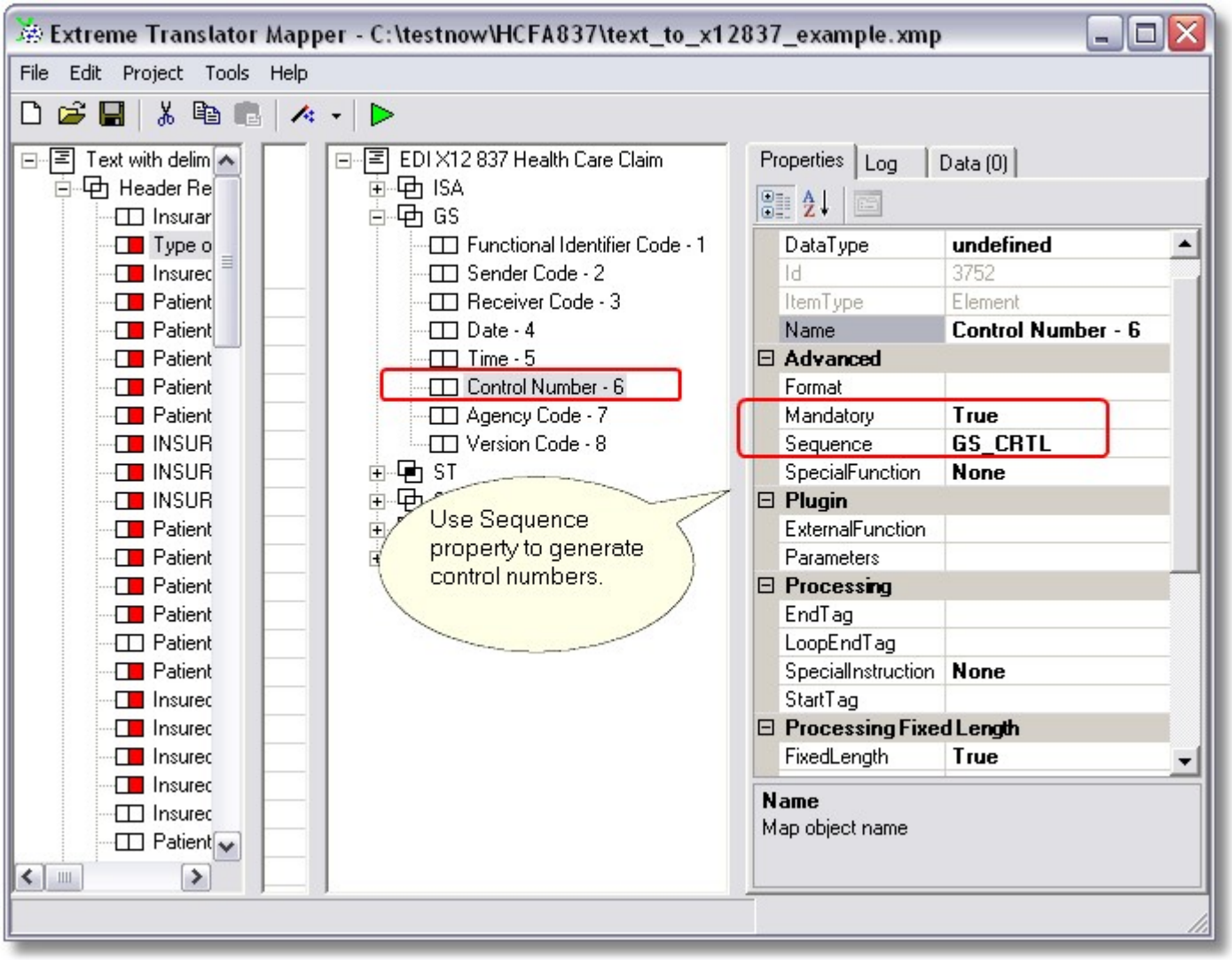



After Template Wizard imports EDI X12 message layouts you should change DataPath property to the output file you want to produce.

If you want to use Sequence property to generate control numbers set IniFile property to the file that will save values used to generate control numbers. So every time translation gets executed new set of control numbers will be generated.

IniFile property is used with Sequence properties to generate and produce control numbers. Because EDI X12 control numbers should increment per each execution of the translation, IniFile property has the name of the file that will hold last generated control numbers, so next time translation runs those numbers would be incremented and new values would be used on each run.

If you decide to reset control numbers to initial value of one, simply delete the file IniFile property points to before you run next translation.



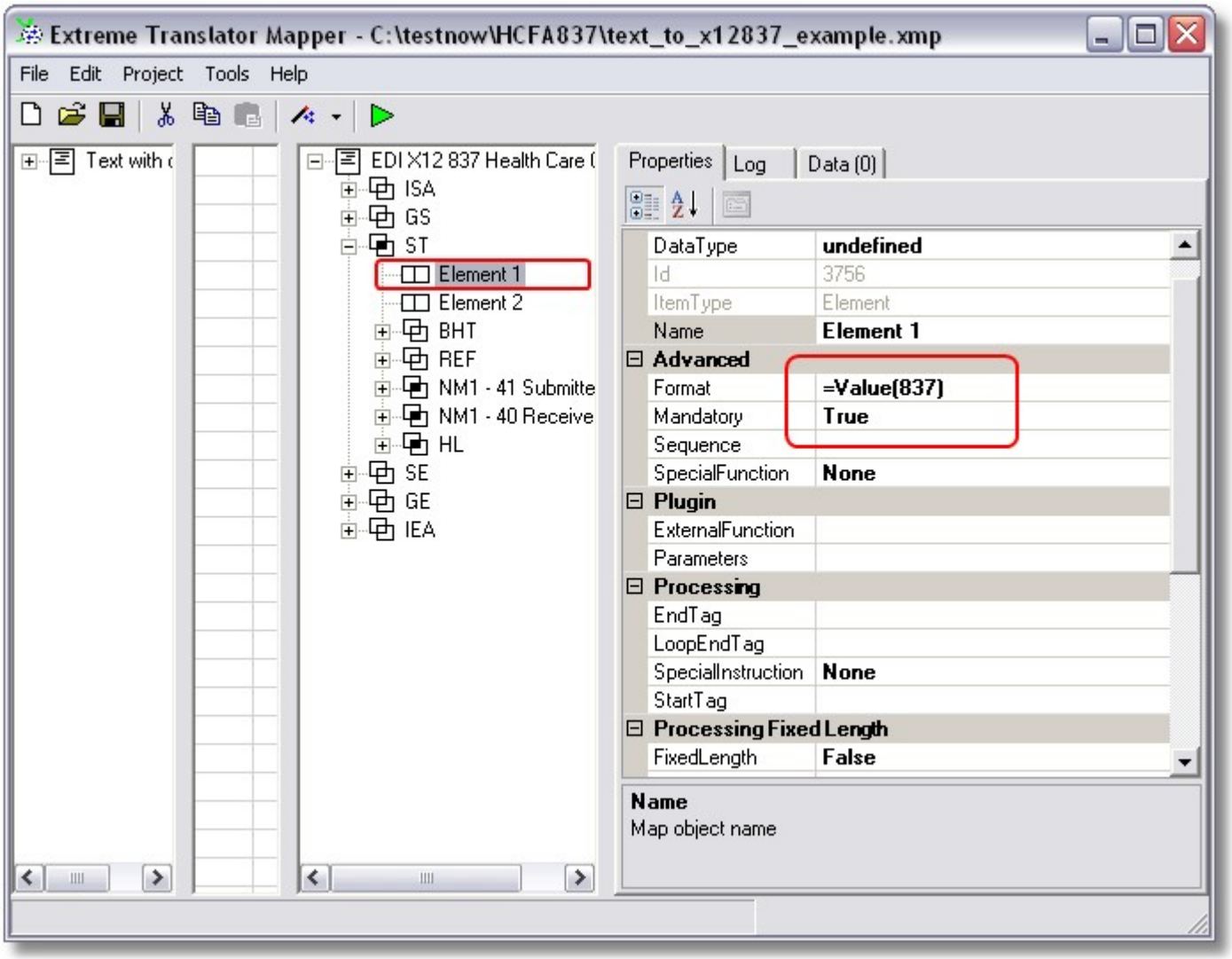

Use Sequence property to produce control numbers on the output side.

If you need control numbers to have fixed length and have zeros padded on the left use FixedLength, Length, PadType and PadCharacter properties.

IniFile should point to the directory that is write enabled. Directories under “C:\Program Files” are read-only on most modern version of Windows, and trying to place files for control numbers there would fail.

**How to produce constant values in the output**

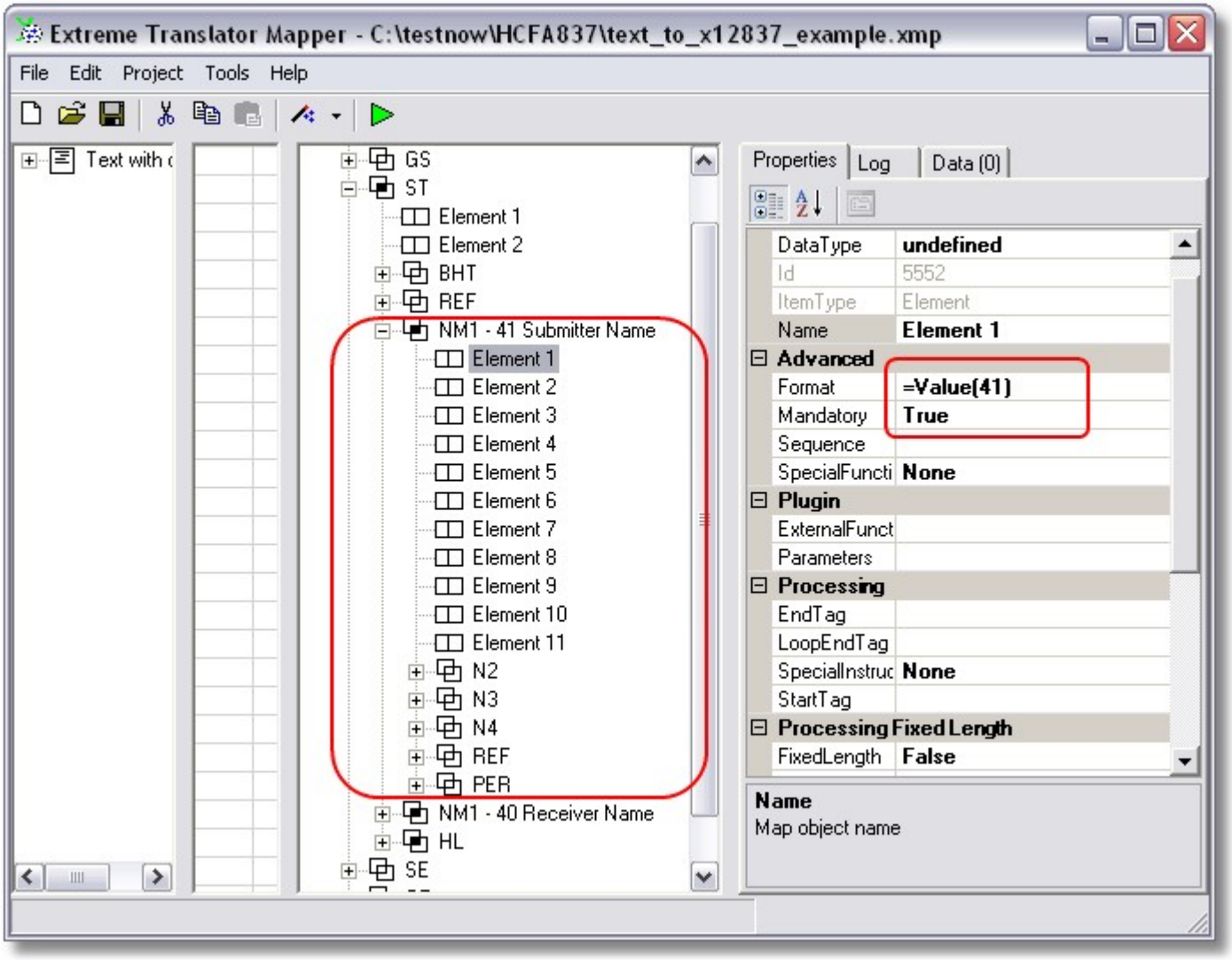
Format property on the output side can be used to reformat data and also to produce constant values in the output.




We can place default values in the output using Format and Mandatory properties. Set Mandatory to true and Format to =Value(your_constant_value).

If segment has no mappings to its nested elements and segments then it will not be produced in the output side. You can force such segment to be produced by setting it to Mandatory=True. If you also would like to output first few elements with fixed constant values make sure to set those elements to Mandatory=True as well.
Basic idea is that Mandatory=True forces segment to be produced in the output even if segment has no mappings inside. But in order to also output elements of that segment you need to set them to Mandatory=True as well.

If you both map elements inside of the segment and also mark segment as Mandatory=True that segment will be produced twice.

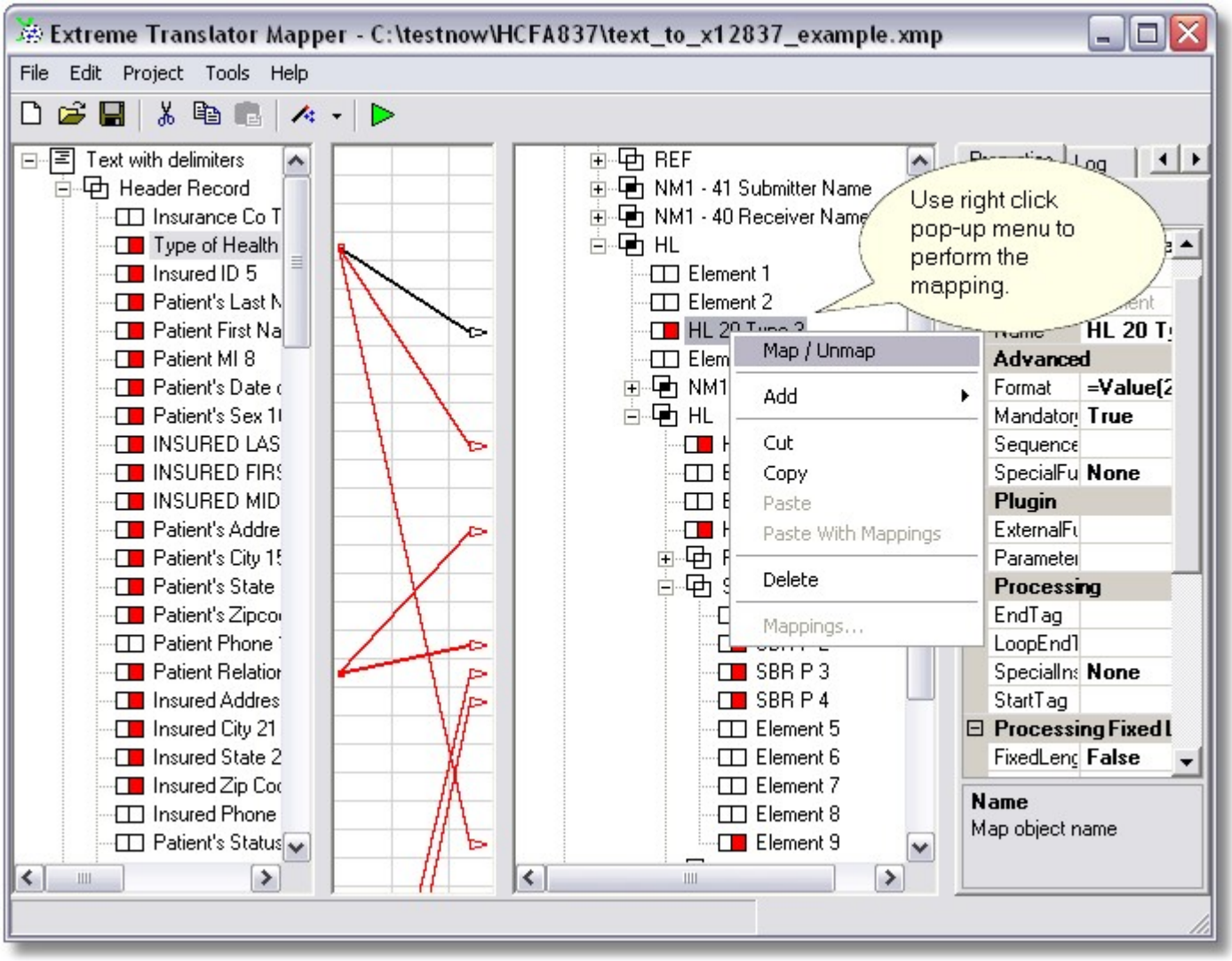



You can also produce number of looping segments. If segment should have different qualifier and output values you can simply Copy and Paste segment or group of segments (entire loop). In this example above we copied NM1 to produce two NM1s with different constant values in them.

**How to do the mapping**

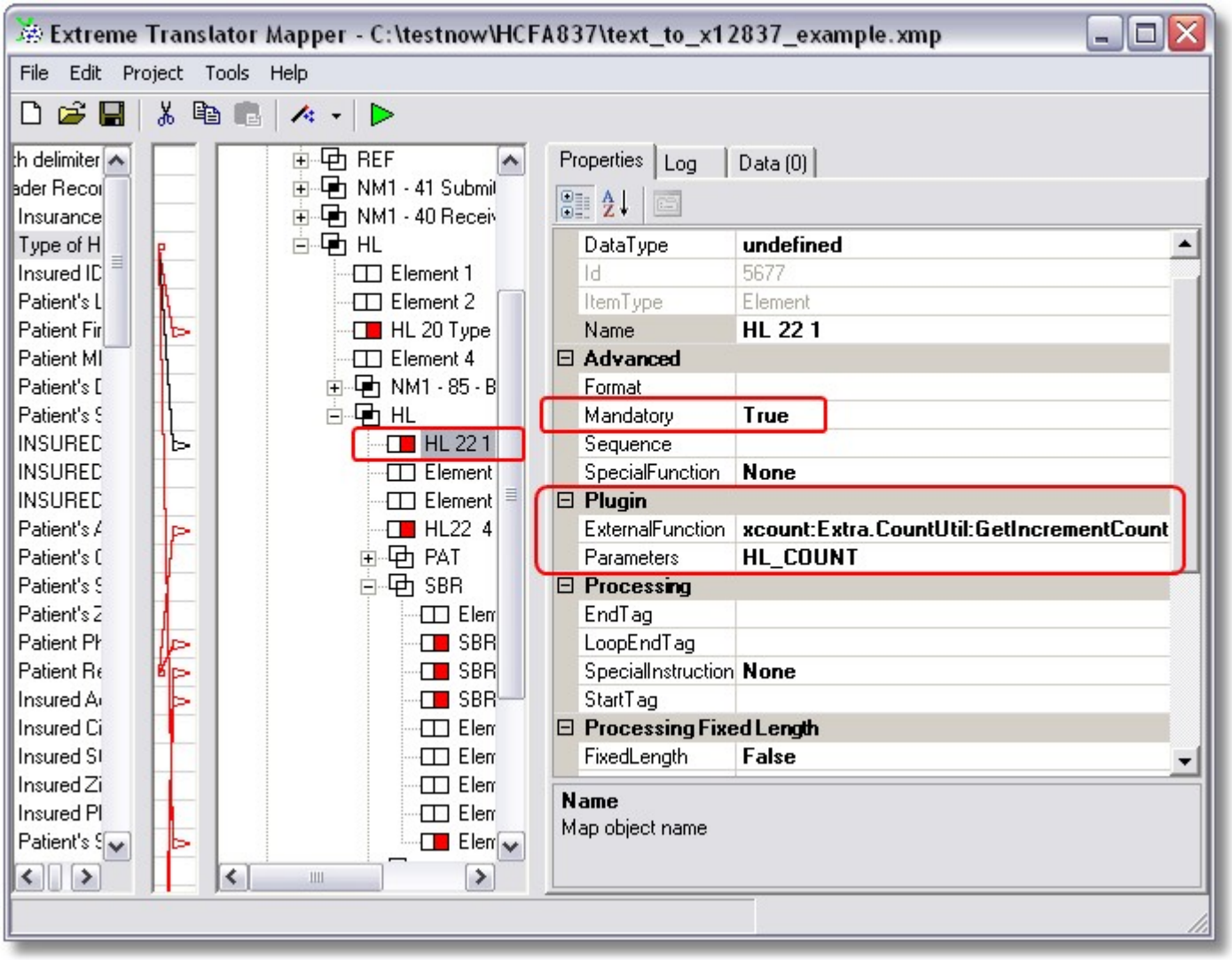
Map does not produce any input data in the output without mappings. Mappings move data from input side into the output. On the output side only EDI X12 elements should be mapped.

In some 837 maps sub-elements also have to be mapped. Please check Users Manual for instructions on how to map to sub-elements.



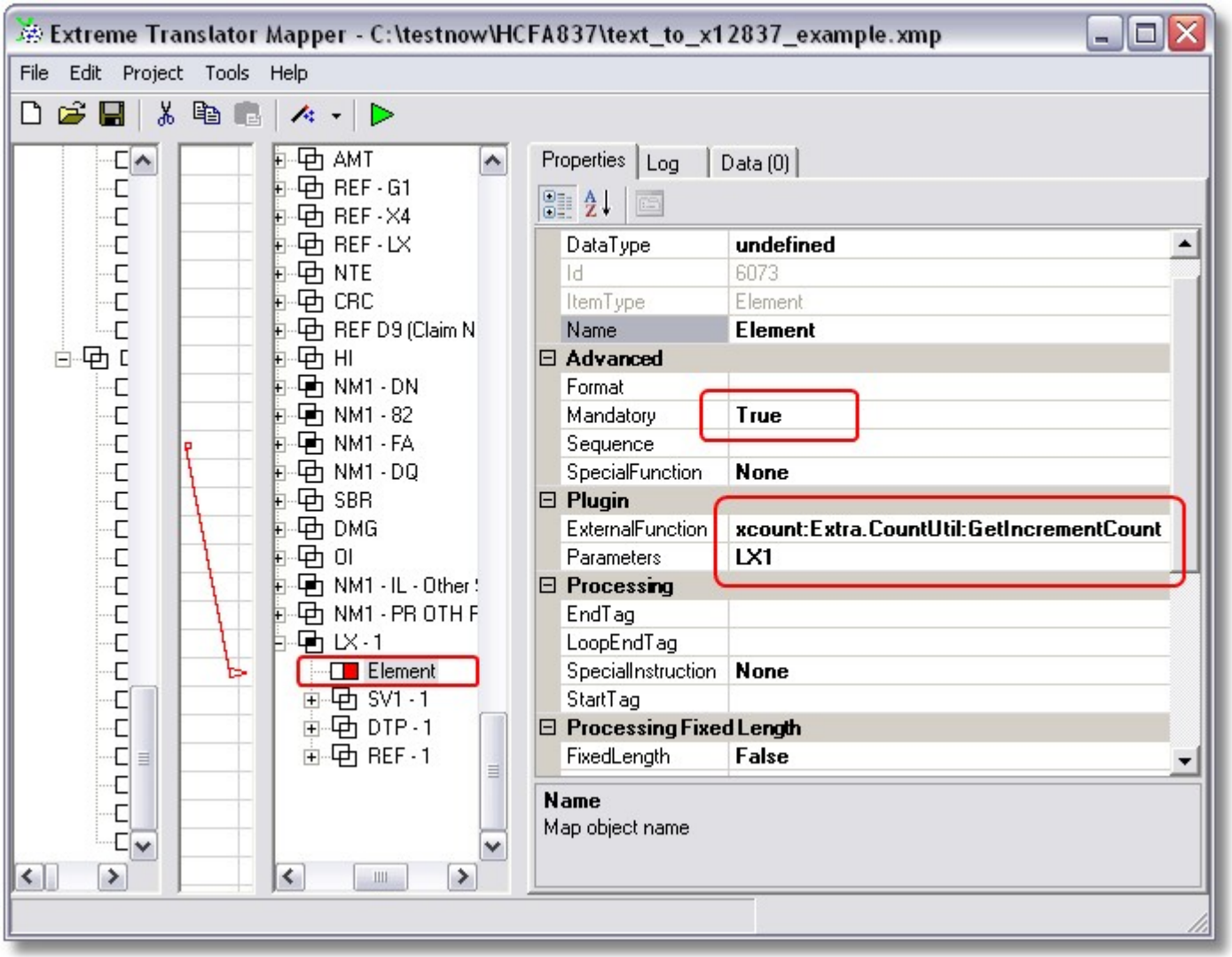

If you want to create mappings you can simply select input, select output then right click on the output side and click “Map/Unmap” pop-up menu item.

**How to create incremental numbers and counts**



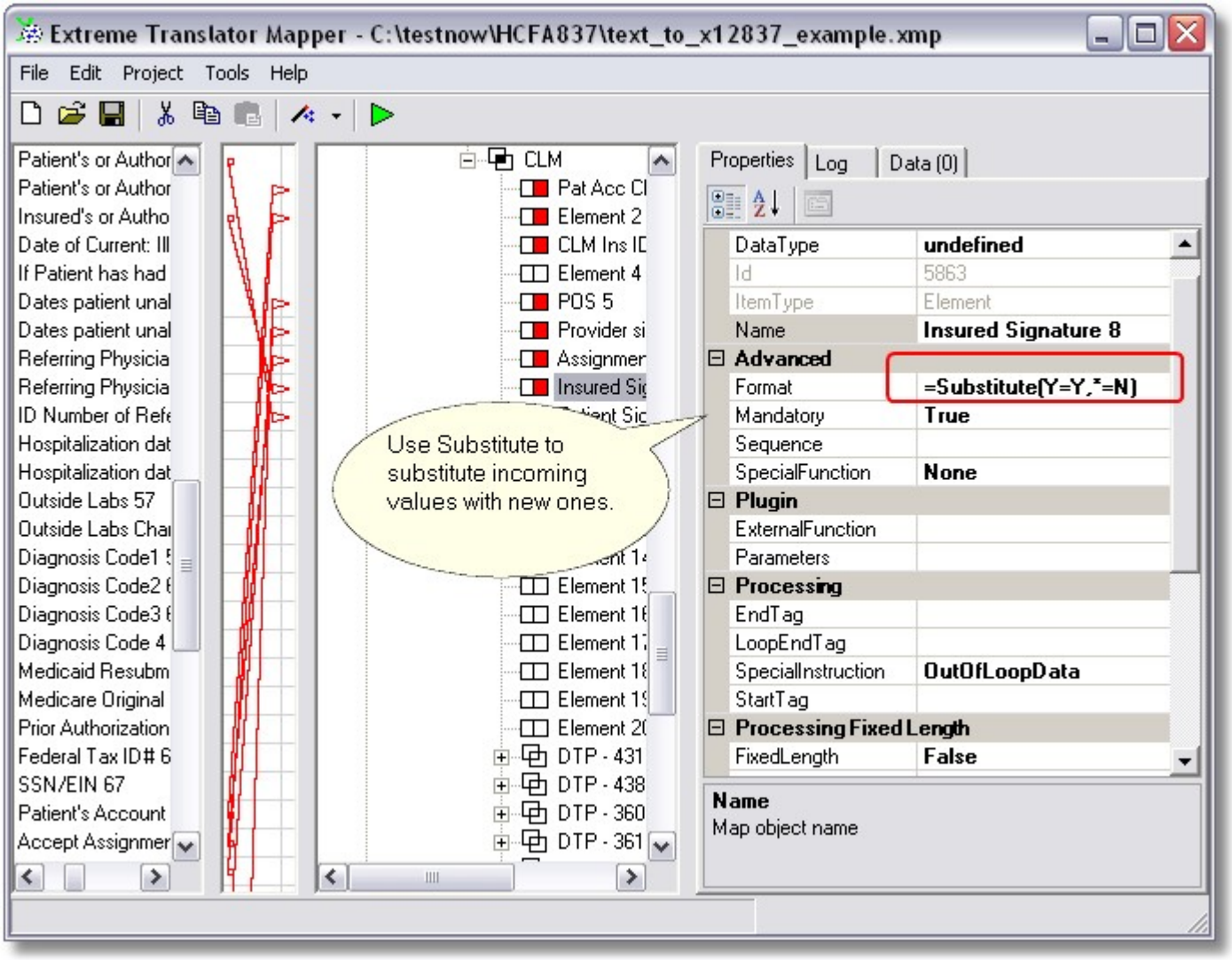

You can set Mandatory to true and use functions that start with xcount: in ExternalFunction property.

xcount functions create, increment or decrement counts. Check plug-ins documentation for complete list of xcount.dll functions.




There is another example of incrementing count. Parameters property is used to name count so it can be reused or reset on other elements.

**How to substitute incoming values**




You can also substitute incoming values by replacing them with new values using Format property =Substitute() function. Star * in =Substitute() function is used to indicate “all other values”.

In the example above =Substitute(Y=Y,*=N) means:

1. If incoming value is Y place Y in the output.
2. If incoming value is something else, then place N in the output.

**Example map**

This document comes with example map “text_to_x12837_example.xmp”. This example is provided just to show basic usage of the various techniques in the translator. The map is not complete and has to be adapted to your specific business requirements.
You can open “text_to_x12837_example.xmp” with Map Editor.