



**XTranslator EDI X12 to CSV flat text file mapping example**

Copyright © 1999-2018 Etasoft Inc.  
Main website <http://www.etasoft.com>  
XTranslator website <http://www.xtranslator.com>

**How to define EDI X12 layout..... 2**  
**How to define flat text file layout ..... 5**  
**How to filter or extract specific EDI X12 segments based on qualifier values ..... 12**  
**Example map..... 16**

## How to define EDI X12 layout

While this document talks about HIPAA X12 837 mapping to flat text file, it can be used to help map any EDI X12 message to flat file. Basic steps are the same for all EDI X12 message types.

EDI X12 files contain loops: repeating segments that may contain other repeating segments. While CSV files usually only have one repeating row. In some cases CSV files may have few rows: header line, detail lines and maybe sub detail lines. But that's rare.

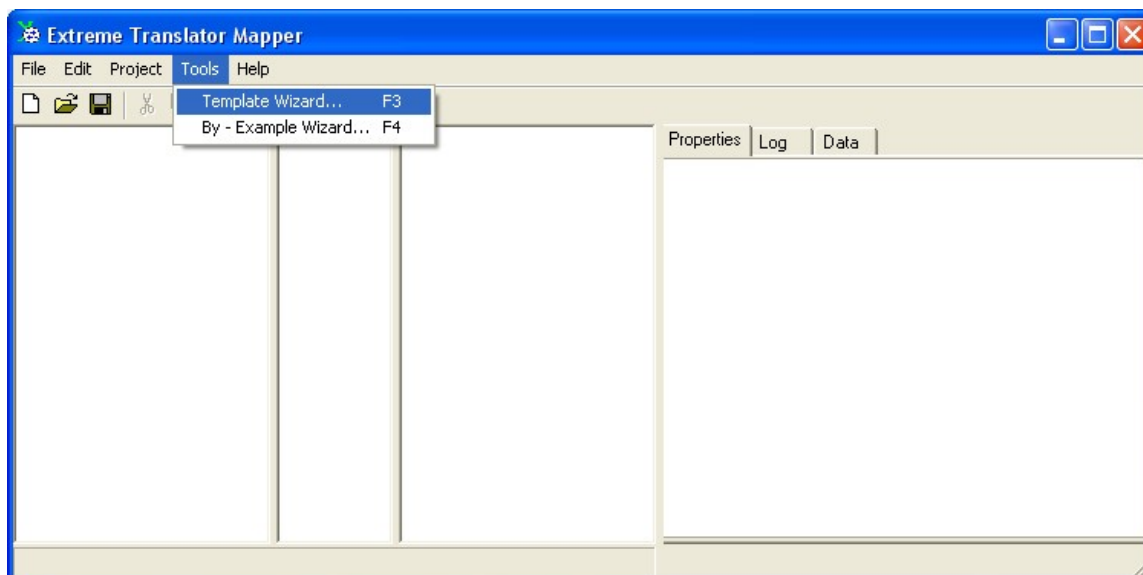
CSV files tend to be flat tables of vertical columns. This makes them perfect for spreadsheet processing or relational databases. This is complete opposite in a way EDI X12 files are structured.

One of the challenges of mapping EDI X12 files into CSV is deciding how to structure CSV file, what fields to choose and how to get EDI loop values repeat over CSV rows.

One complex EDI X12 map may not produce desired CSV. You may have to create two maps and run each map on the same EDI X12 file to get two CSV files with different set of output fields. Another alternative is to create one CSV with header, detail and sub-detail rows to represent different EDI loops.

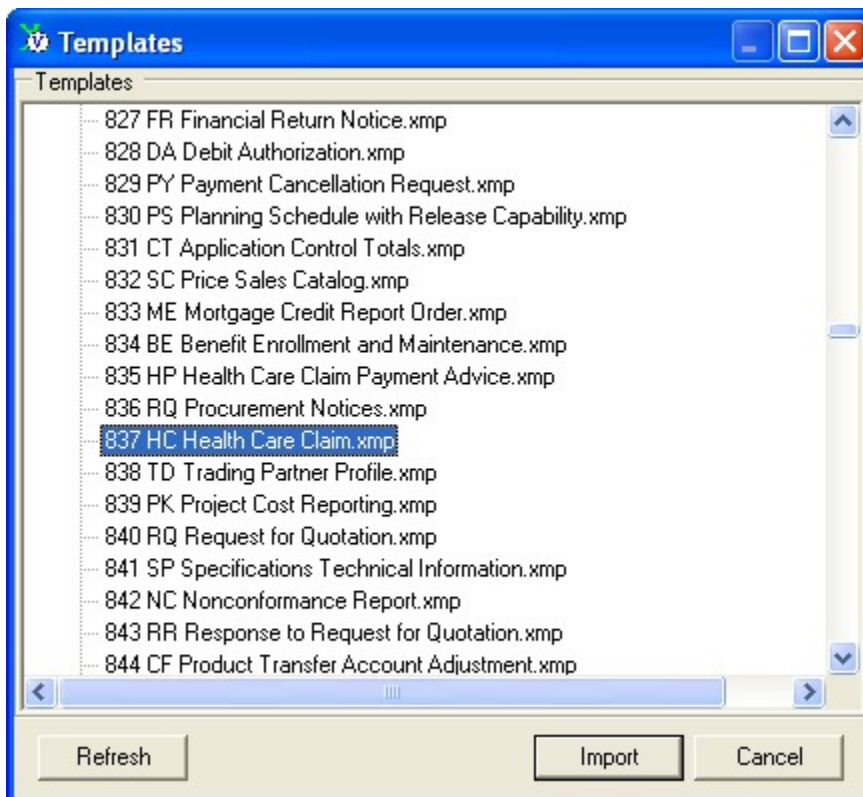
**This document comes with complete map in a file with extension xmp. Please open and examine it as you read this document.**

In this example we use pre-built template for EDI X12 837 and setup CSV file layout manually.

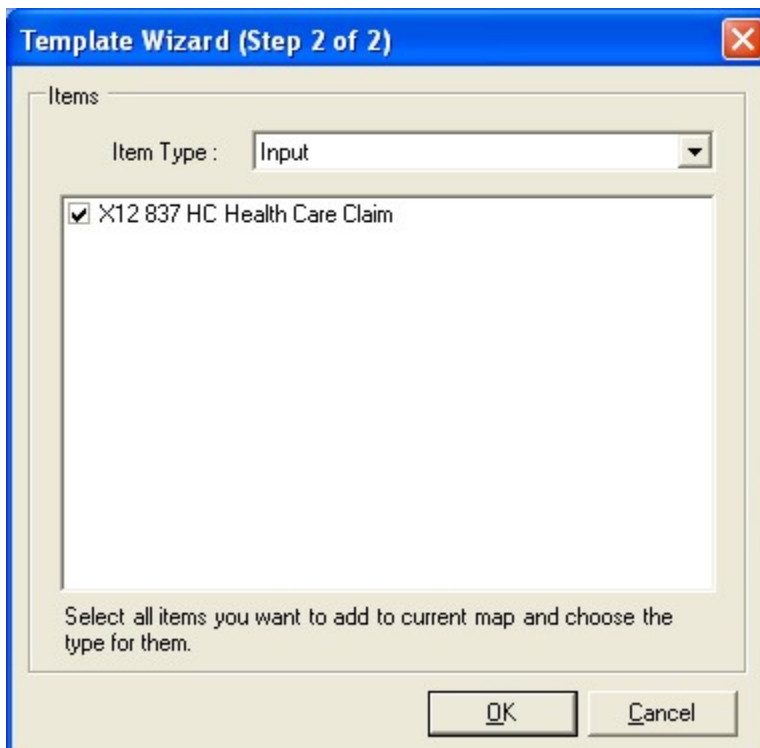


We start with Template Wizard.

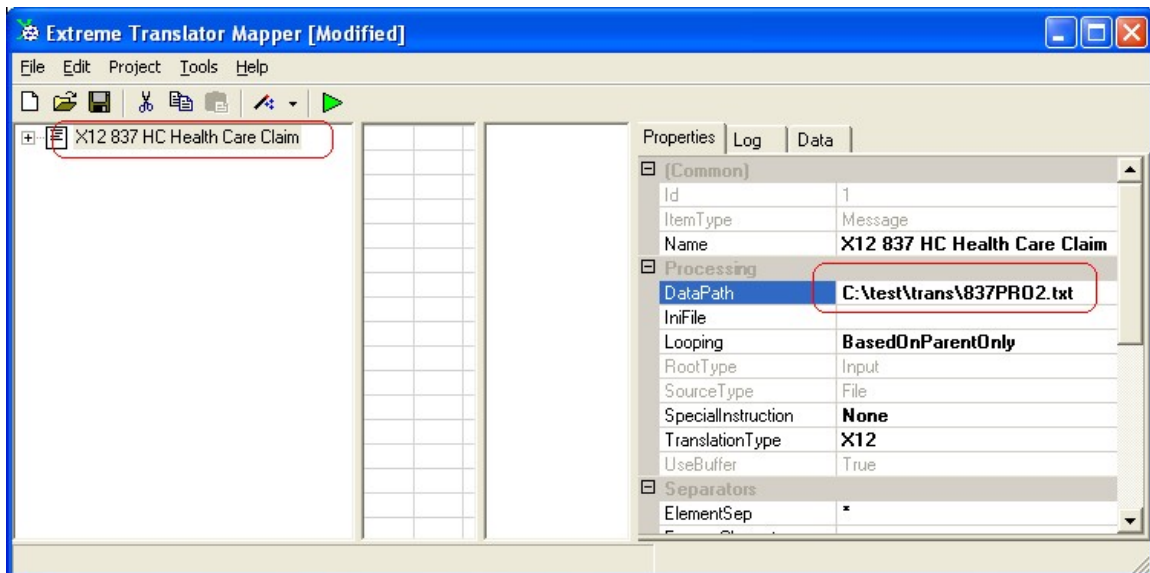
Template Wizard has collection of pre-build templates. There are over 5300 templates based on EDI X12 and EDIFACT standard. All templates are grouped by release date or version number.



We choose pre-build X12 4010 message 837 Health Care Claim. Click “Import” in this dialog. If you want to map some other EDI X12 message simply choose one from this list.



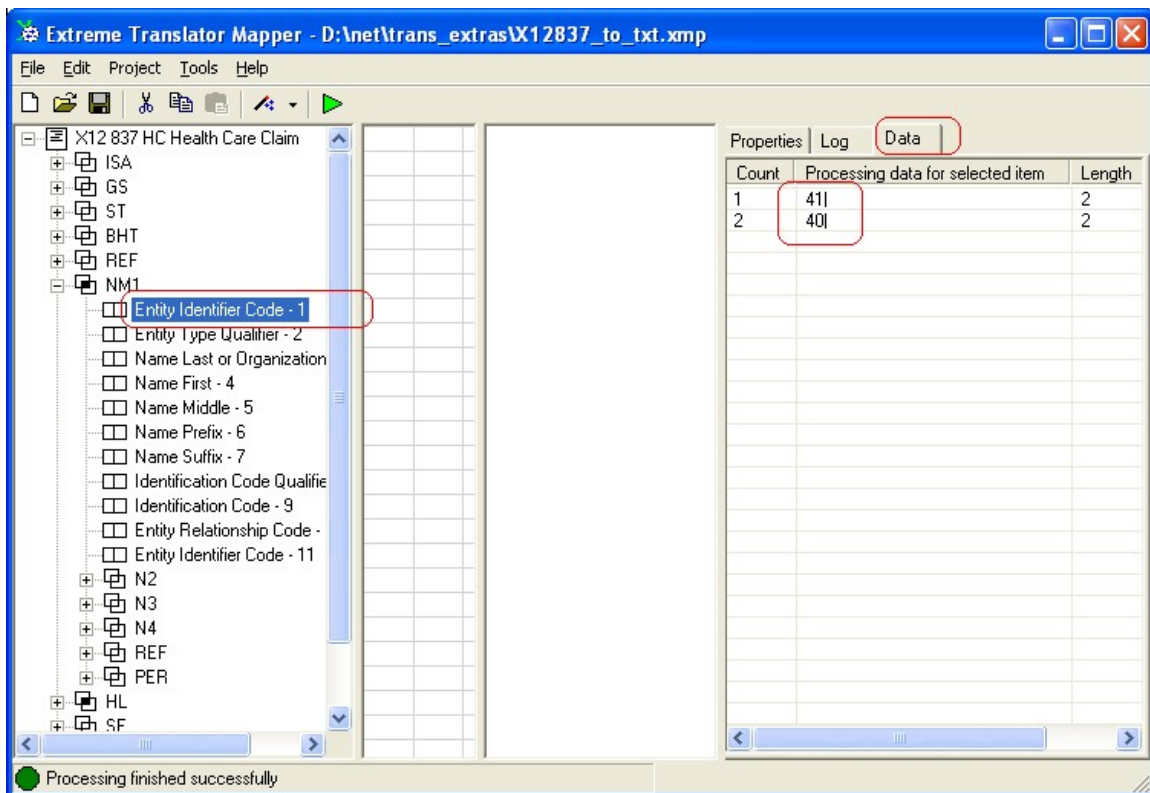
It is "Input" file as we are going from X12 837 to CSV text file.



Select imported EDI X12 template.

Template got imported, and we update DataPath property to read input X12 file. **Input side DataPath should point to existing EDI X12 file.**

After that we can run the map. Translator will read the file and try to parse it. If everything is OK then it will be without warnings.



Processing finished successfully, click on any item in left pane and select “Data” tab. You will see actual data from input file.

**Data tab is important for development.** If you run map at least once Data tab shows actual “live” data of selected map item. You can click on any map item and inspect its contents. Repeated values displayed on separate rows.

Use Data tab often both on input and output side.

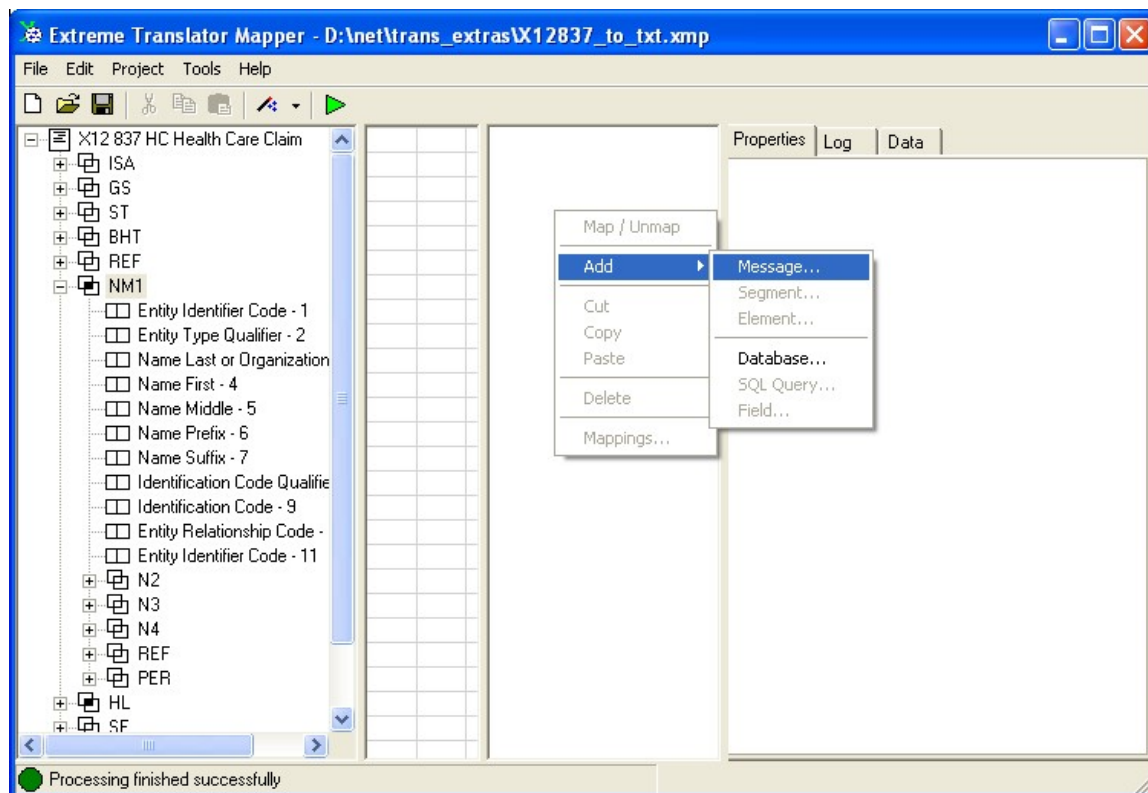
**Log tab shows processing warnings.** If input data does not match segments set in the map you will get warnings displayed in the Log tab. Warnings signal problem with the map that has to be fixed during development time. Do not use map with warnings in production.

Number of warnings may not match actual number of problematic segments in the map. You may get multiple warnings for one segment if segment is the first segment of the loop, multiple segments will get reported because nested segments will get flagged as warnings too.

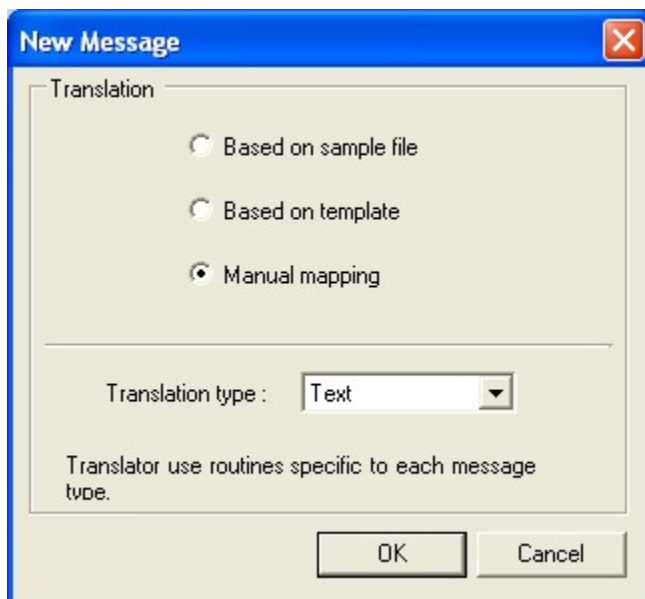
At this point we are done with input side. Sometimes you may need to adjust Separators (click on root item of the tree to see these properties), but in this case translator detected X12 file separators automatically.

### How to define flat text file layout

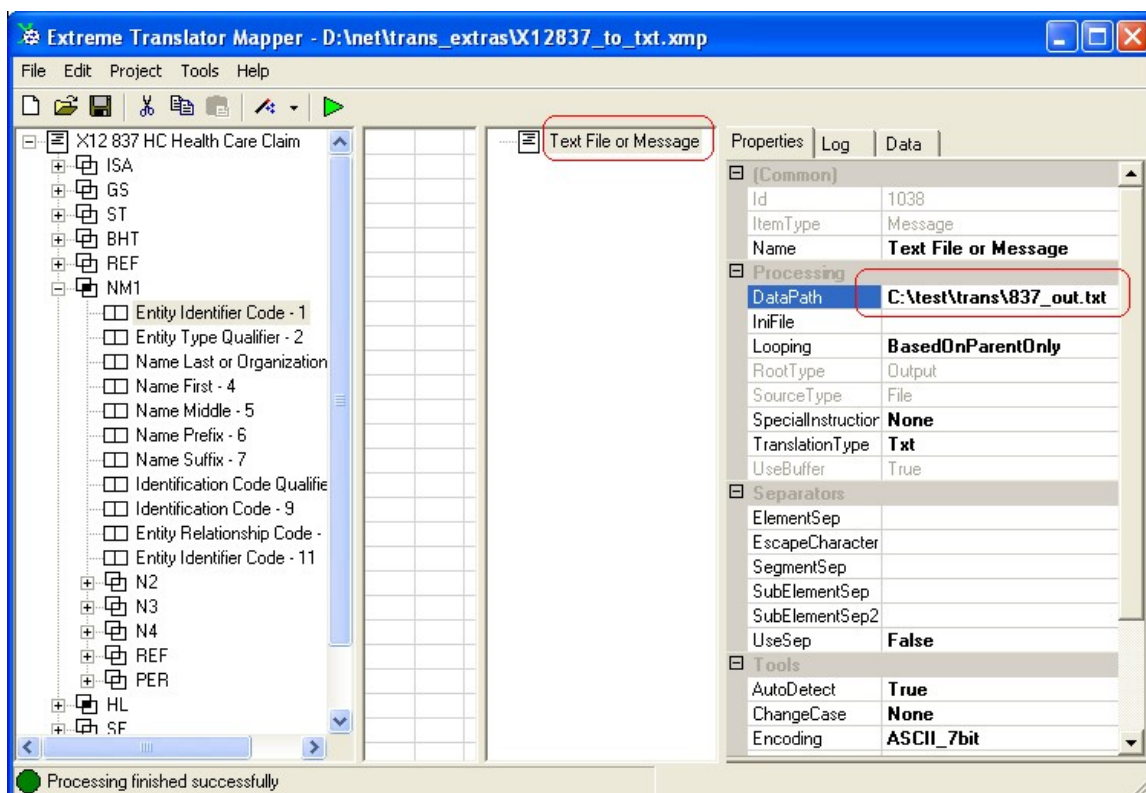
In this example we setup CSV file manually. We add output message first. Then add segment next. Segment in CSV file represents single repeating CSV row with multiple fields (elements).



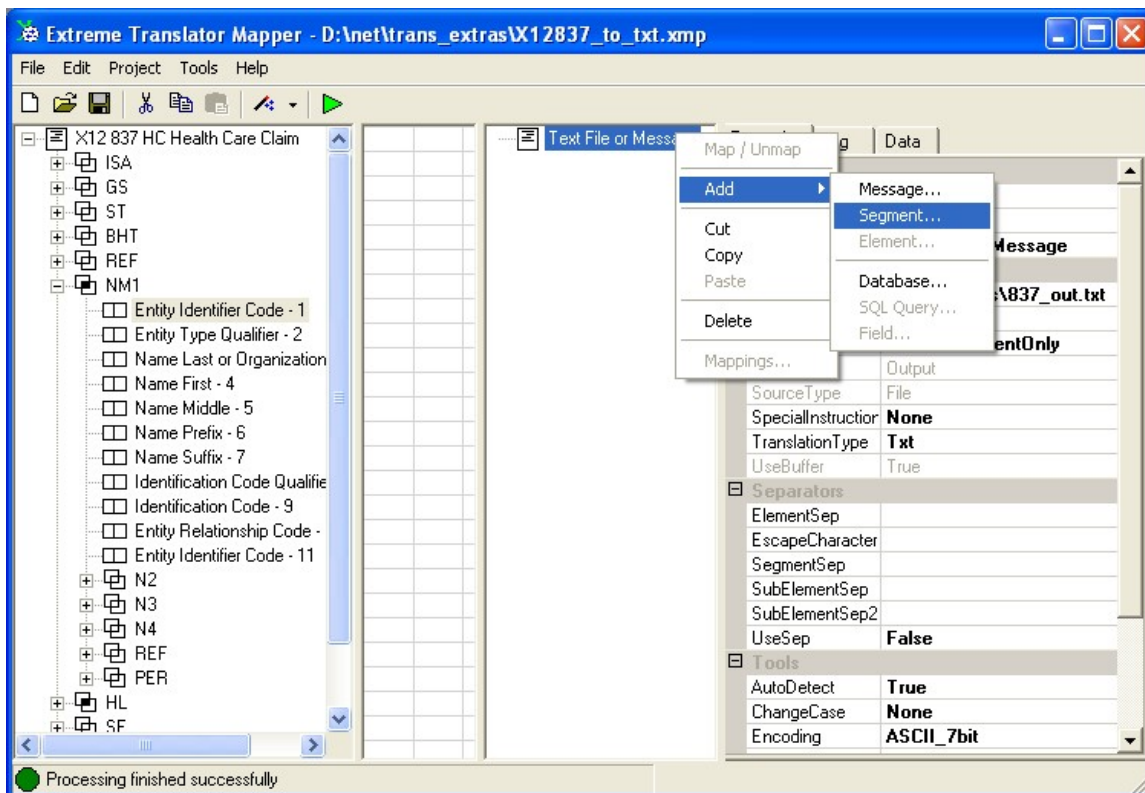
We right click third pane and choose to add Message.



It is going to be our output CSV file.



We enter output file name for text message DataPath property.



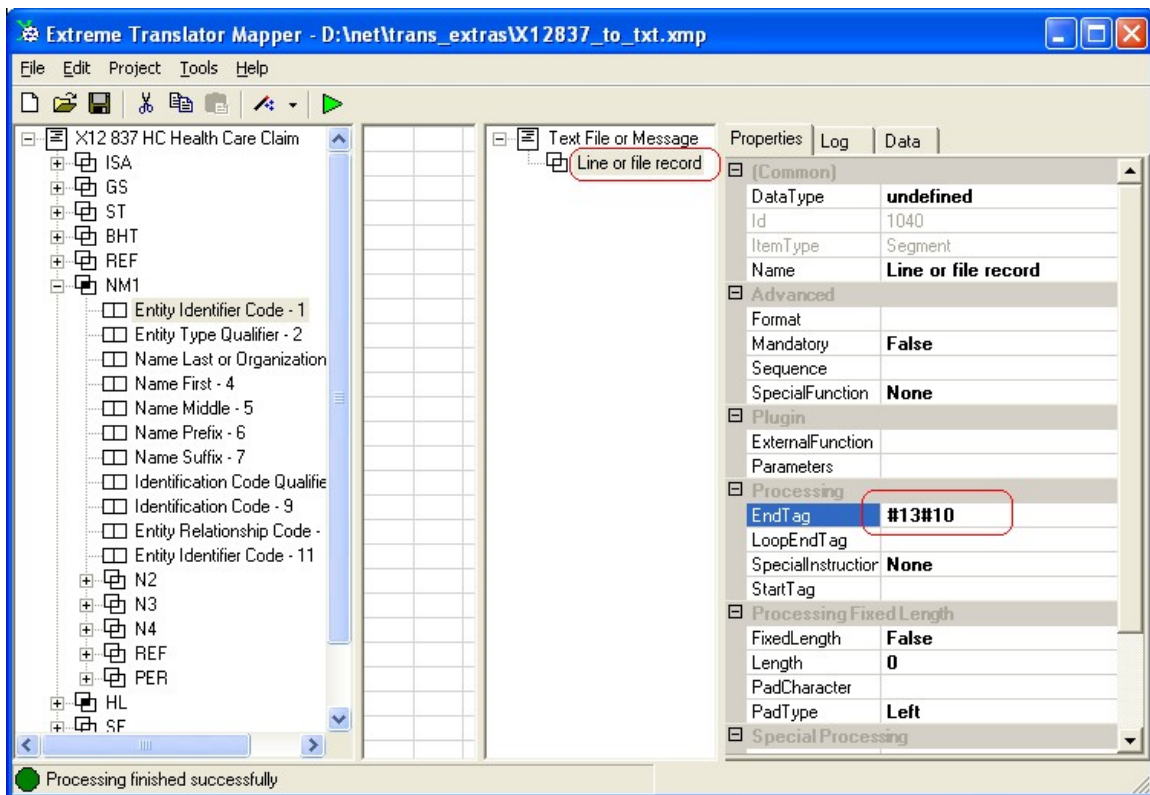
We add segment to the message. Segment represents single repeating line in output CVS file.



We choose name for the segment.

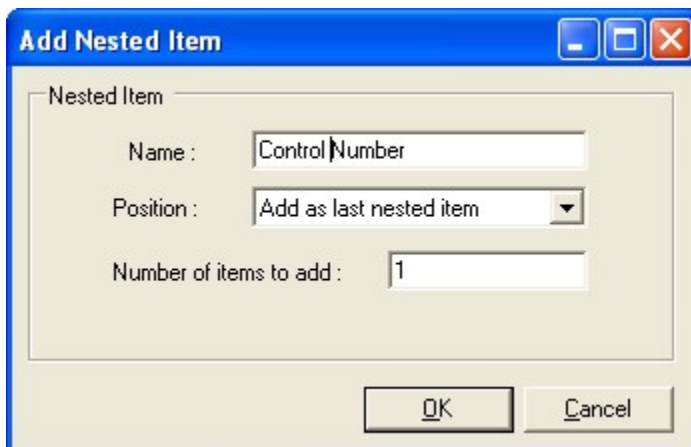
Segment name is used for convenience to be displayed in the Map Editor. It is not used for processing. If you want each line to start with specific prefix use StartTag property. **StartTag and EndTag properties are essential for segments and elements.**





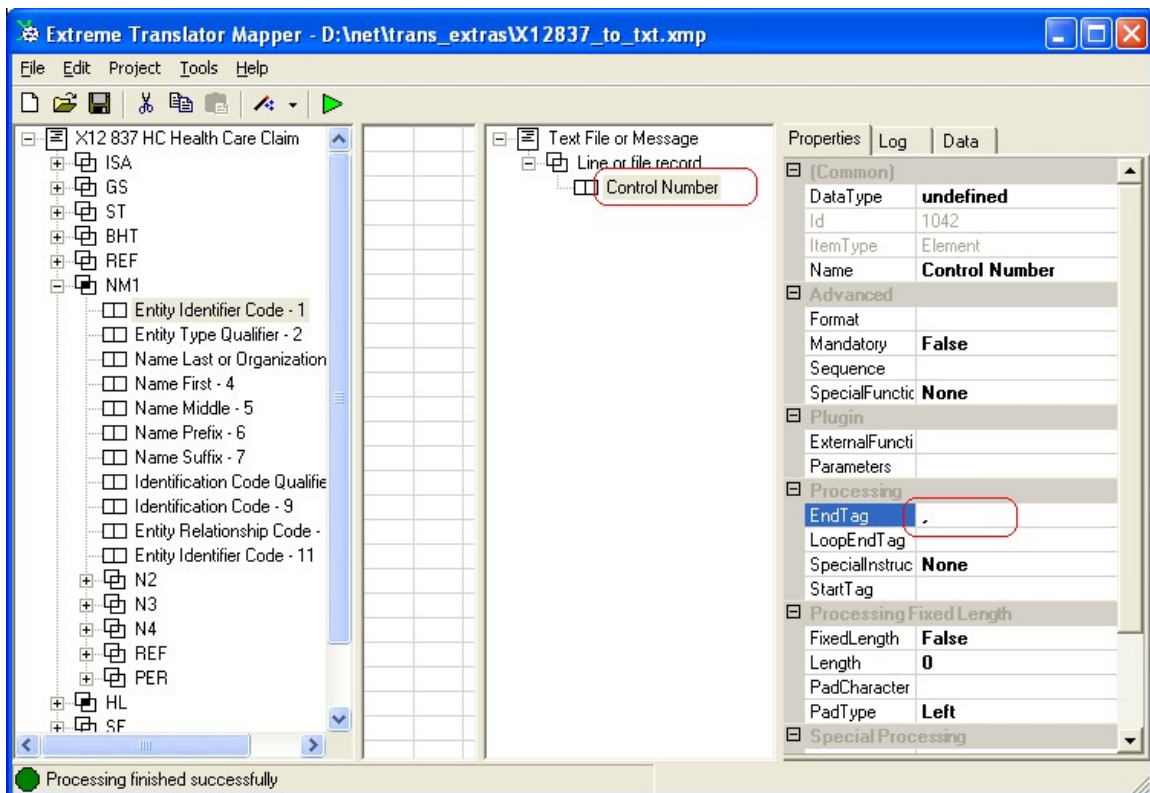
Each line has to end with carriage return and line feed characters.

We enter carriage return and line feed characters using special escape character #. Decimal code has to follow this special character. Carriage return and line feed is represented as #13#10 in the Map Editor screens.

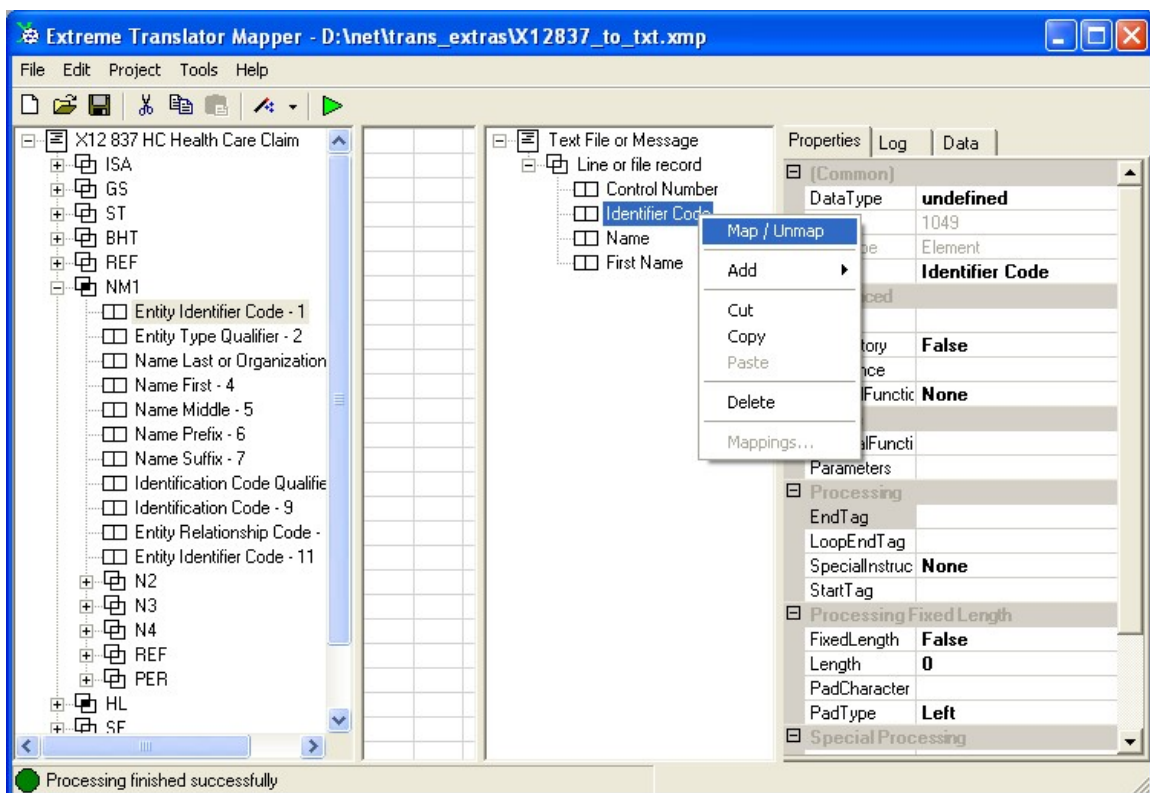


We also add elements to the segment. Element is a single field in CSV file.

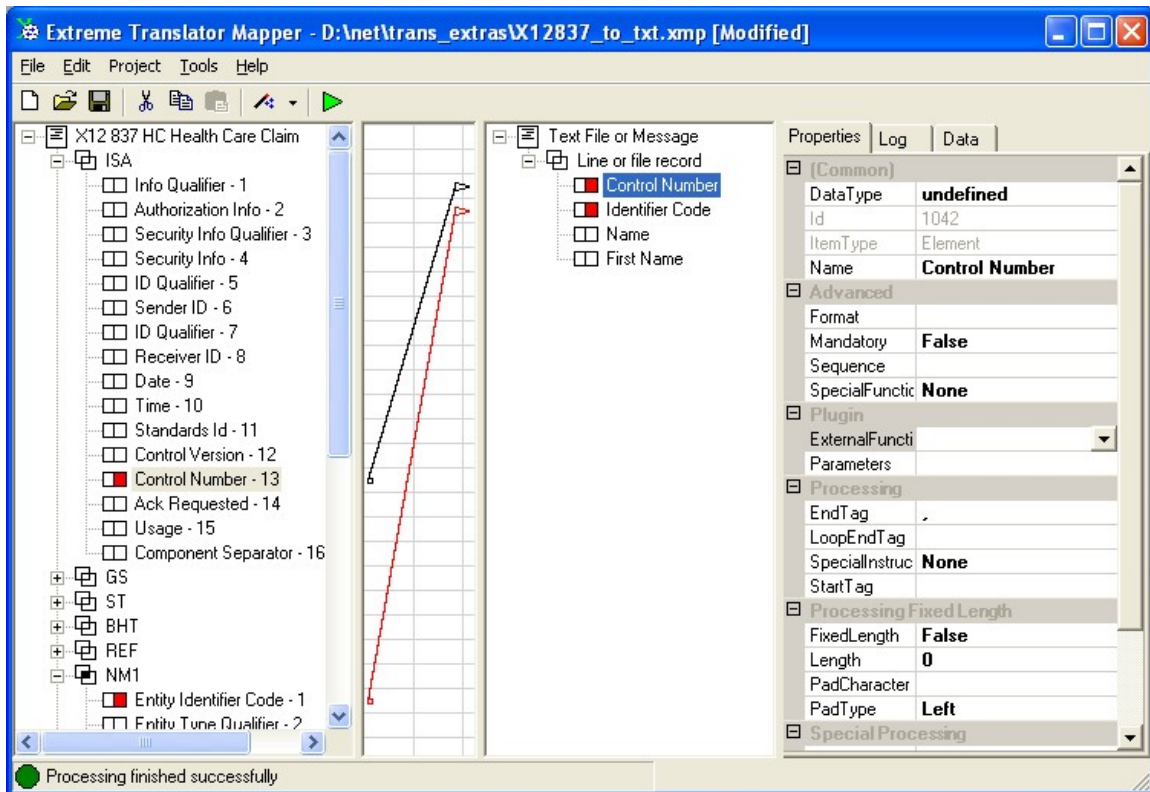




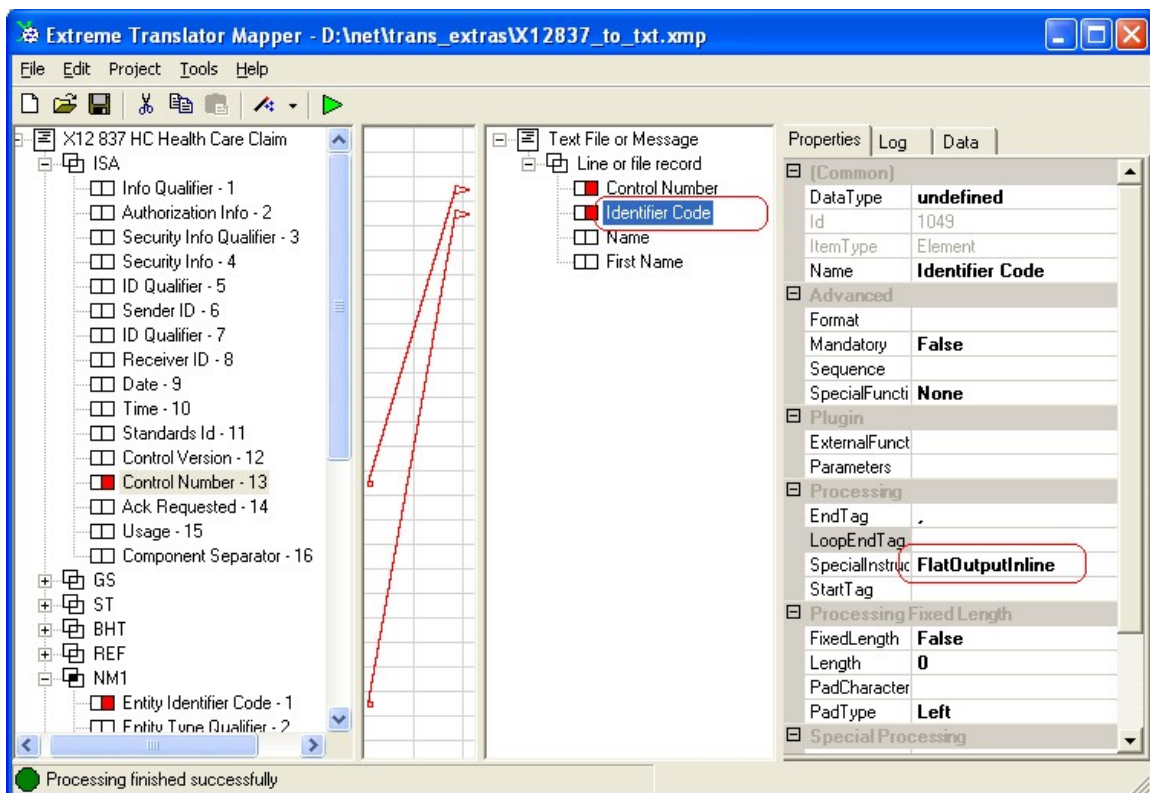
Comma should be placed in EndTag property for each element as it is separator for CSV files.



Once more elements are added we can map them to input X12 file elements.



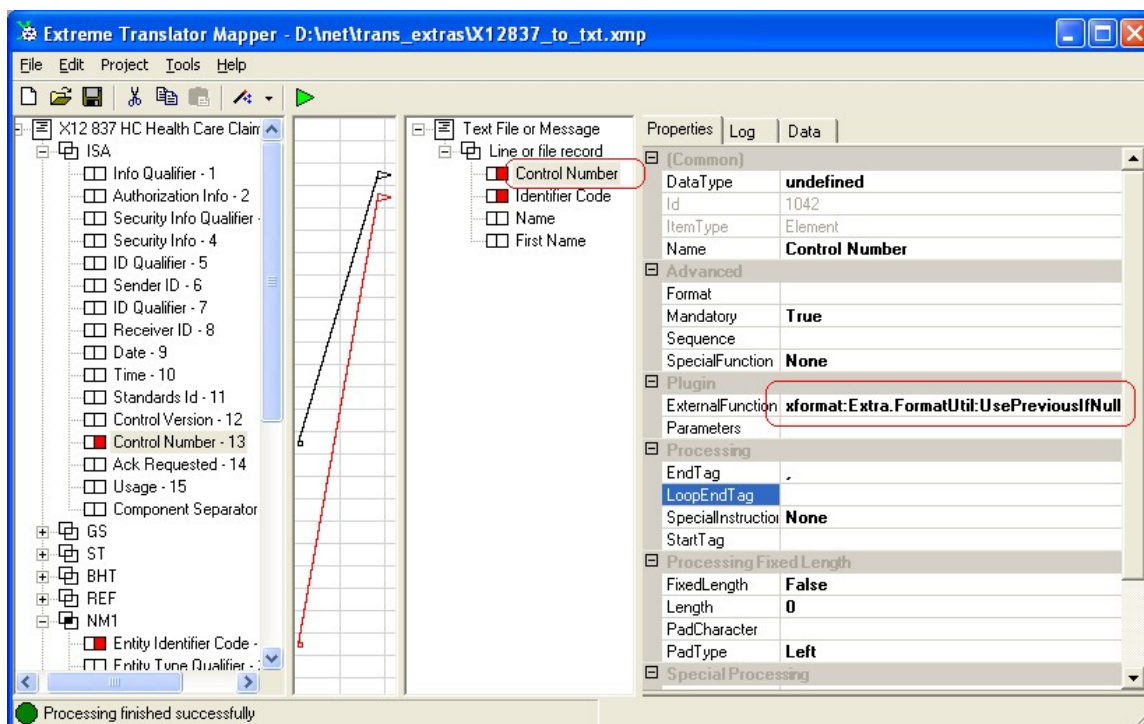
Mapped elements displayed red in the Map Editor.



Because of EDI X12 loops translator may produce CSV file with output data shifted between lines.

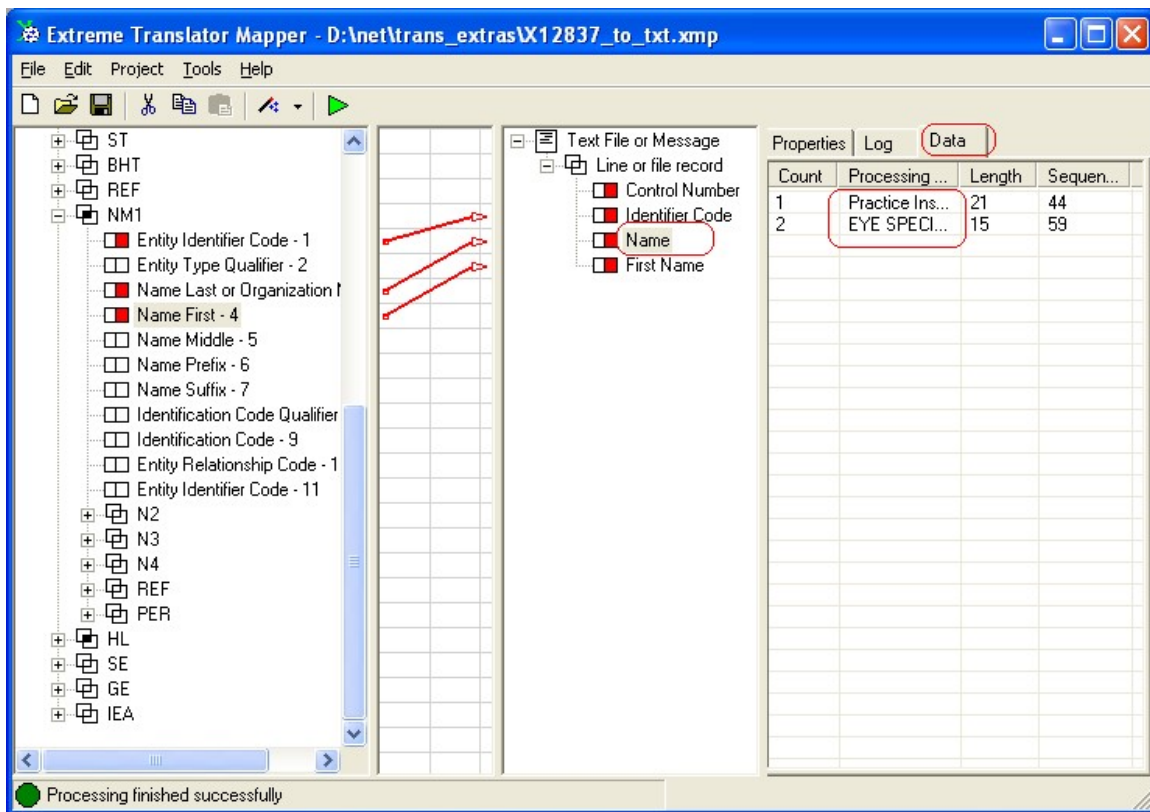
There is a way to hint translator on how to line up items correctly. One of output elements has to be marked as field that is used to line up all other elements.

Best element for this role is one that is always present in the input side for each row of the CSV file. Typical example is EDI X12 detail line item from SV1 or SV2 segments. It is usually first item of the detail loop in X12. We choose "Identifier Code" as primary key. Element is marked with FlatOutputInline for SpecialInstruction property.



Control Number element is only present once per our file but NM1 is looping.

We would like to have Control Number repeat on each line in CSV text file. This is done using ExternalFunction set to xformat.Extra.FormatUtil:UsePreviousIfNull. You can use this function to repeat number of header fields that only get few values from the input file.



This is view of the output file for element Name.

### How to filter or extract specific EDI X12 segments based on qualifier values

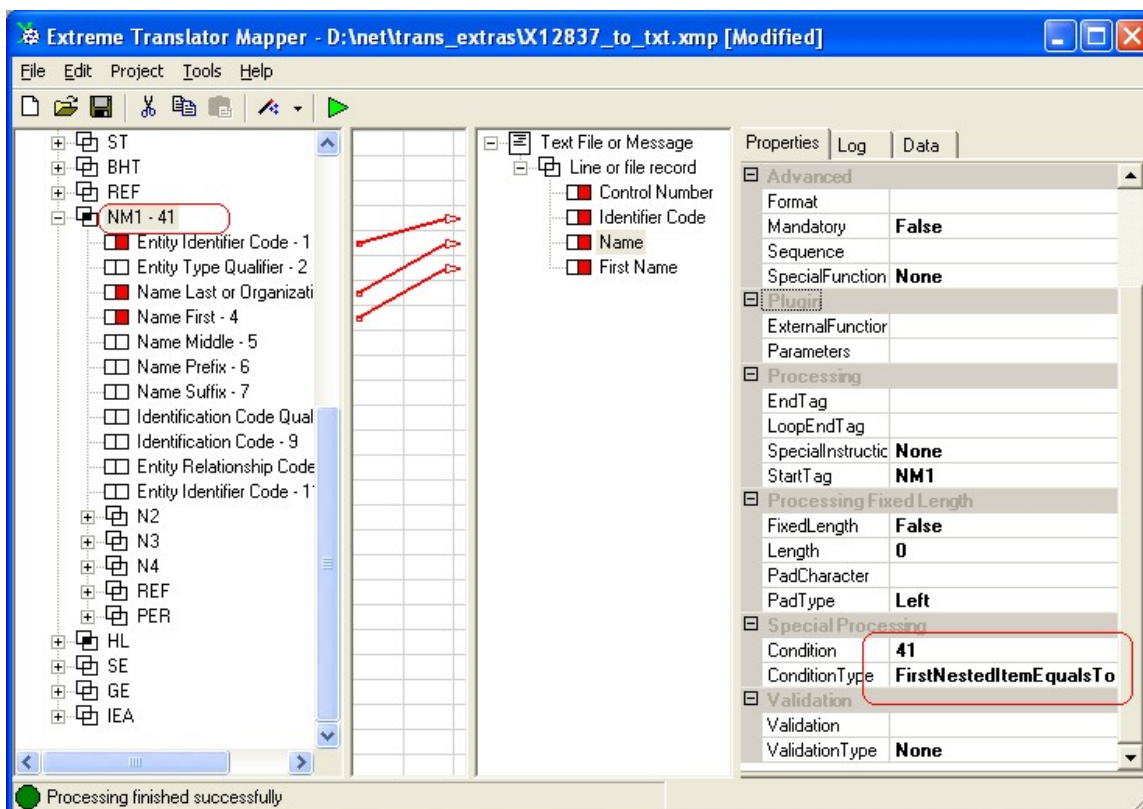
You can use Condition and ConditionType properties to filter specific EDI X12 segments or route them to other flat file fields based on values in EDI X12 elements. This feature is mostly used to perform certain translations based on so-called EDI qualifier values.

Qualifiers are specific elements in EDI X12 segment that tell us what that segment or group of segments actually means.

Example: N1 segment usually has element #1 as a qualifier, and for example if element #1 contains "PR" that means N1 and segments below (N2, N3, N4, etc.) contain payer address information. Qualifier defines what type of address is in N1, N2, N3, N4 segment group (loop).

Qualifiers are used extensively in EDI X12. In cases when you need to filter specific segment or segment group, set Condition and ConditionType properties on that segment. Example: for N1 with qualifier PR in element #1 set Condition to PR and ConditionType to FirstNestedItemEqualsTo.

NM1 is another segment that usually has element #1 containing qualifier value.

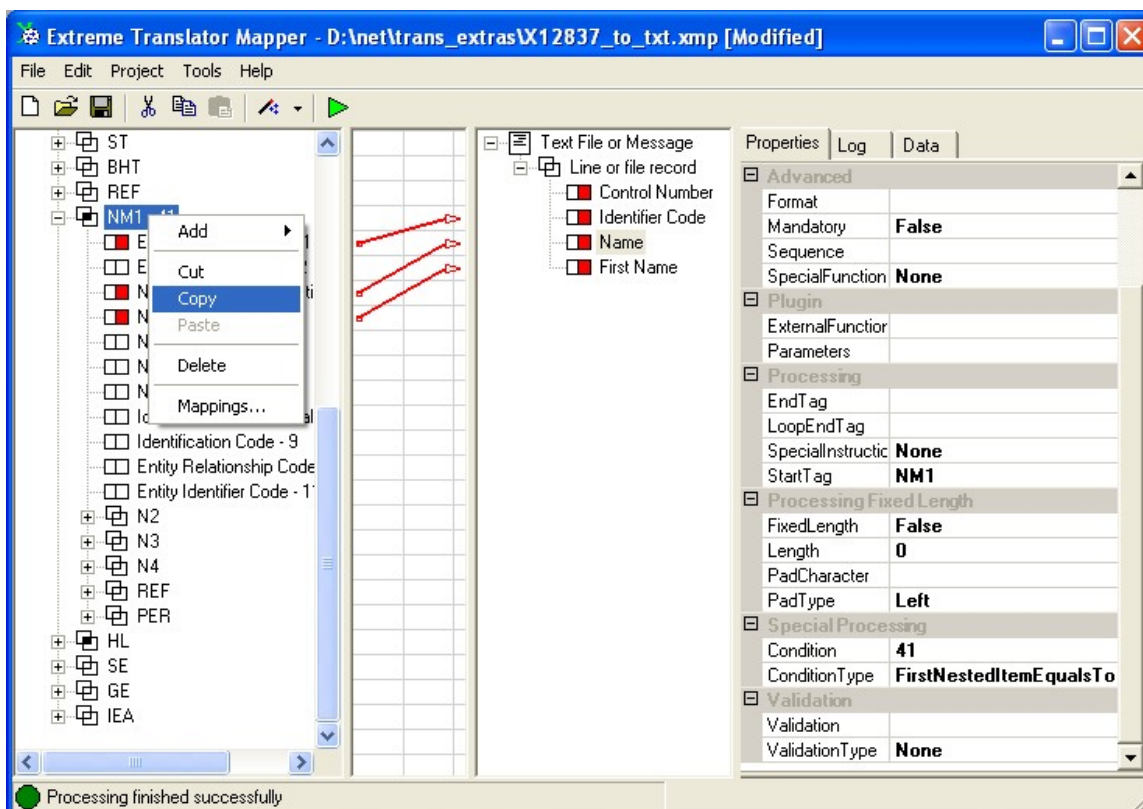


We have two NM1s on the input.

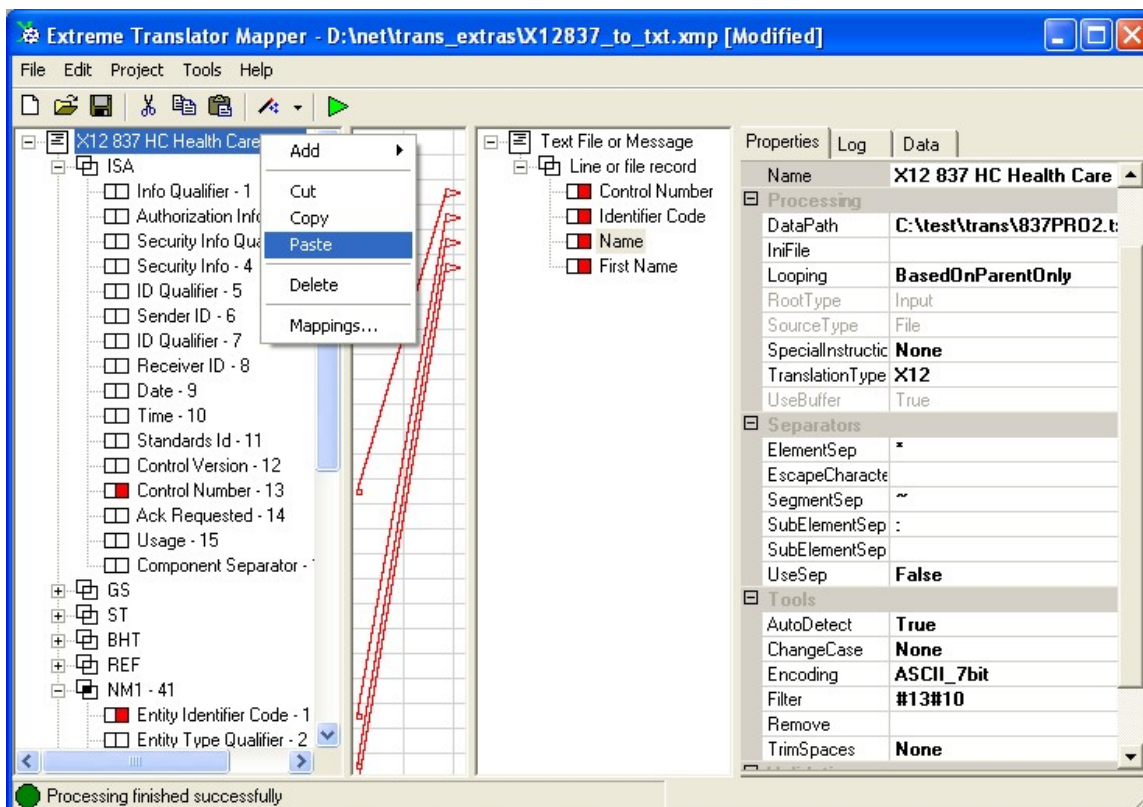
One NM1 has qualifier (first element) as 41 and another NM1 has 40. We are only interested in NM1 with qualifier 41. It is easily done using Condition and ConditionType properties on the segment.

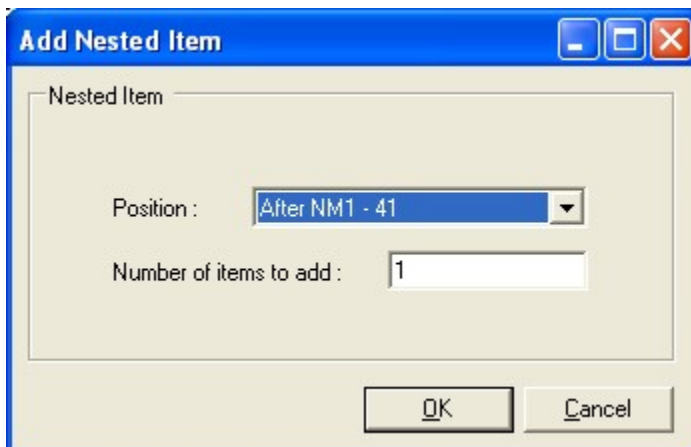
We click on NM1 without any Condition/ConditionType set, right click on it and press Copy. Then select parent of that NM1 segment and choose menu Paste. We paste new NM1 below existing NM1. After these steps we have two NM1s at the same level. Now we can setup unique Condition value on each NM1.



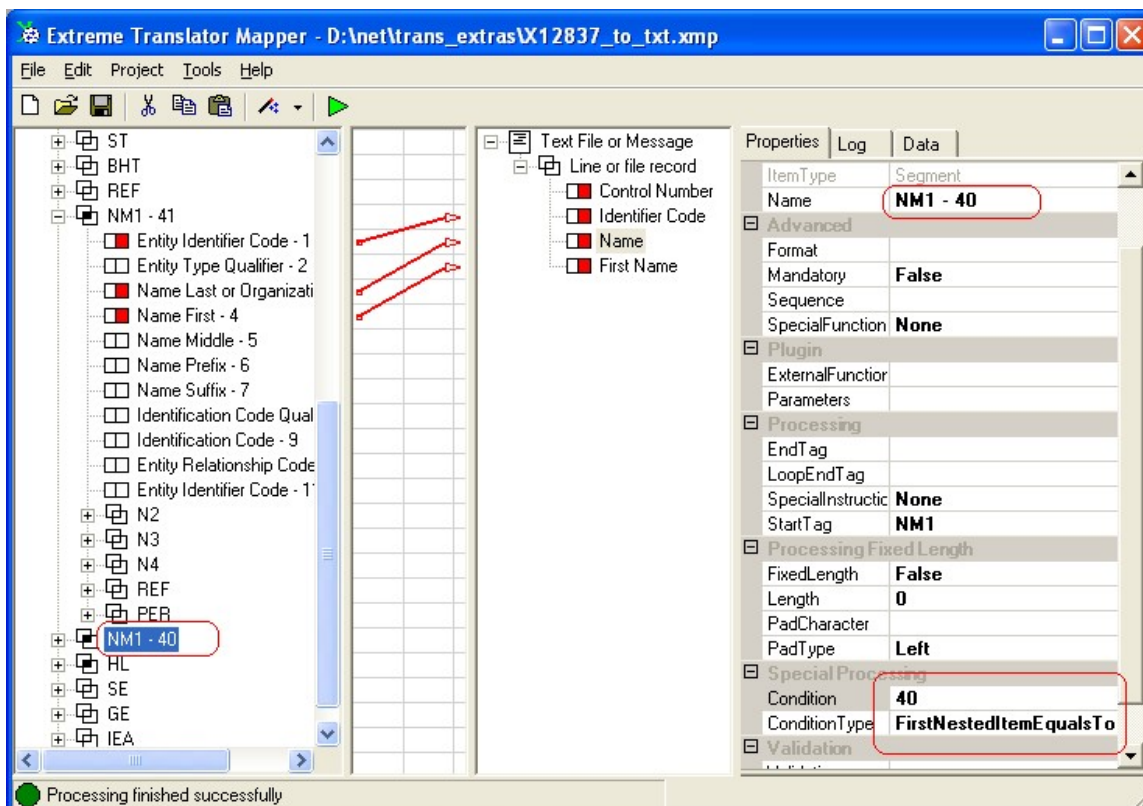


Now we want to read NM1 segment with qualifier 40 as well. So we copy current NM1, and paste it under.





We paste copy under existing one.



Now we could map "NM1 – 40" to some other elements in CSV file and continue.

NM1 without any Condition/ConditionType captures all qualifiers at that specific loop level. Once you setup multiple NM1s with different Condition/ConditionType you may get warnings in the Log tab.

When Conditions are setup on the segment make sure you capture all incoming qualifier values. Let say you have setup NM1 with qualifier "FA" and NM1 with qualifier "77" but



you also receive NM1 with qualifier “78” at that loop level as well but you have not created third NM1 with Condition set to “78”.

In this case Missing NM1 with Condition set to “78” will cause warnings to show up in the Log tab.

### **Example map**

This document comes with example map “X12837\_to\_txt\_complete.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 “X12837\_to\_txt\_complete.xmp” with Map Editor.