XTranslator looping issues explained

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

Purpose

Purpose of this document is to explain issues about mapping looping data and some advanced techniques such as mapping database queries to form header and detail lines on the output files, mapping repeating nested X12 EDI segments to flat output text files, mapping X12 EDI to XML, etc.

Looping Introduction

XTranslator is a sequence based translation engine. It tries to order data in the output based on the sequence of data in the input. There are ways to alter that behavior if need be. However most of a time it is a desired behavior. But sometimes it is not that simple to order data when formats of input and output are very different. Sometimes data might be missing in the input side and throw sequence off, that may shift ordering on the output in undesirable way.

Example: ordering looping EDI X12, EDIFACT or XML messages going to flat files could be very hard just because data formats are conceptually different. X12, EDIFACT and XML messages have loops or looping tags that may repeat many times when flat files in most cases have only one loop (only one line). What is worst is that some input loops may repeat 10 times, some may repeat 100 and some are present only under some conditions, when in flat file they should always be there even if there is no data in the input, at least as place holders for flat file to keep its structure, also loops that repeat may shift output flat file. Other scenario could be situation when data from relation database has to be translated to X12, EDIFACT or XML loops or vice verse loaded into database tables. Tables are like flat files. That's why it is so easy to import or export them from relational databases using standard tools. However in the map you would probably want to join few tables in order to form header and details in X12, EDIFACT or XML files.

Looping issues when mapping X12 or EDIFACT to XML

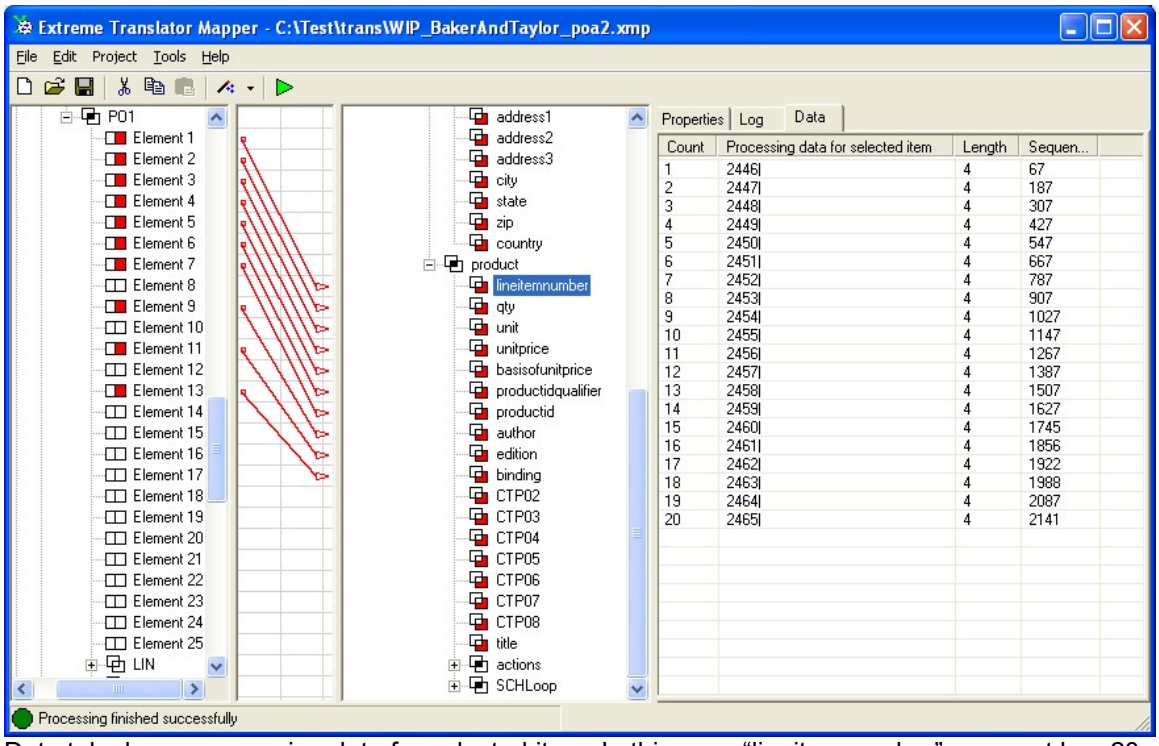
While both X12/EDIFACT and XML structures are similar. They both have loops and segments containing other segments, looping issues can still appear on XML side in cases when:

1. X12/EDIFACT data is missing on some of the loops
2. XML tags are nested very deep.

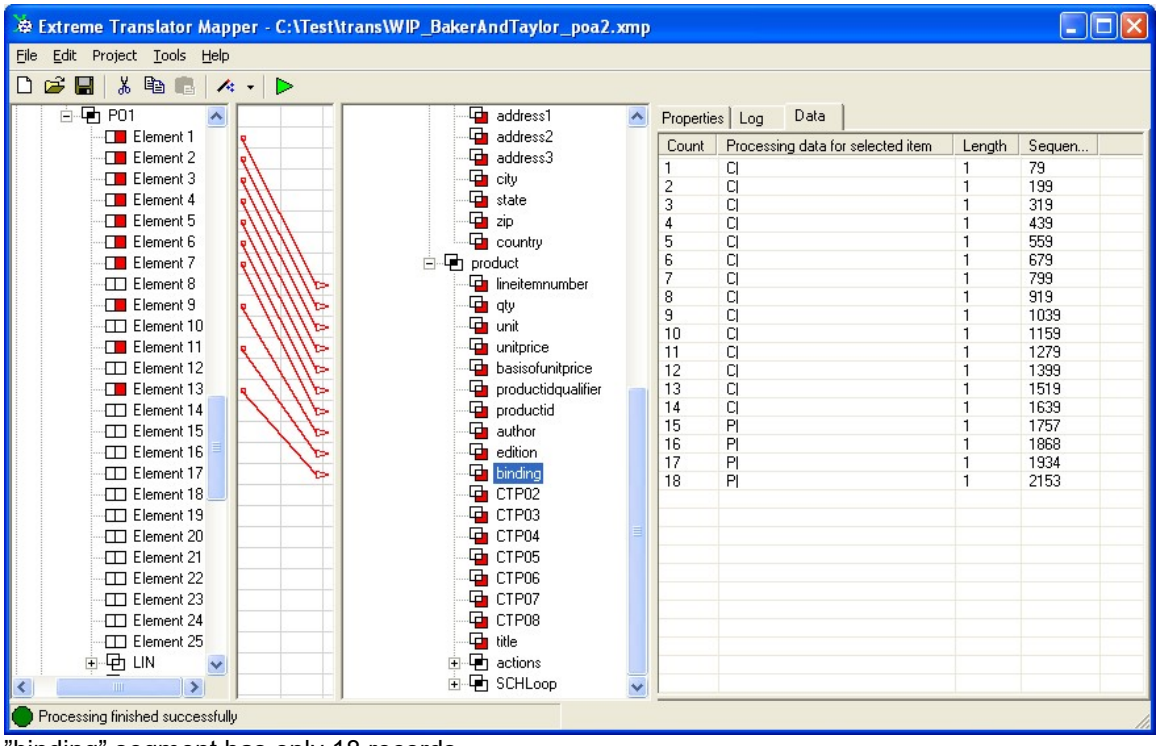
1. When X12/EDIFACT data is missing in the XML segment that is in the group and some other segments are nested under that group. Example:

```
<order>
  <stcontrolnumber></stcontrolnumber> (always present in input)
  <ponumber></ponumber>  (always present in input)
  <transactionpurpose></transactionpurpose>  (let say sometimes it is not present in the input)
  <product>
      <some_product_detail></some_product_detail>
  </product>
</order>
```

In this example <product> tags could get ordered incorrectly in the output because <transactionpurpose> sometimes is missing on X12 side. You can use SpecialInstruction property on "transactionpurpose" segment and set it to OutOfLoopData. That way translator will know that some data might be missing and will determine sequence based on other segments. Actually it will use segments "stcontrolnumber" and "ponumber" with SpecialInstruction = None to order data in the output.
If you do not know what tags should be marked as SpecialInstruction = OutOfLoopData you can easily find out by using "Data" tab.



Data tab shows processing data for selected item. In this case “lineitemnumber” segment has 20 records.



”binding” segment has only 18 records.

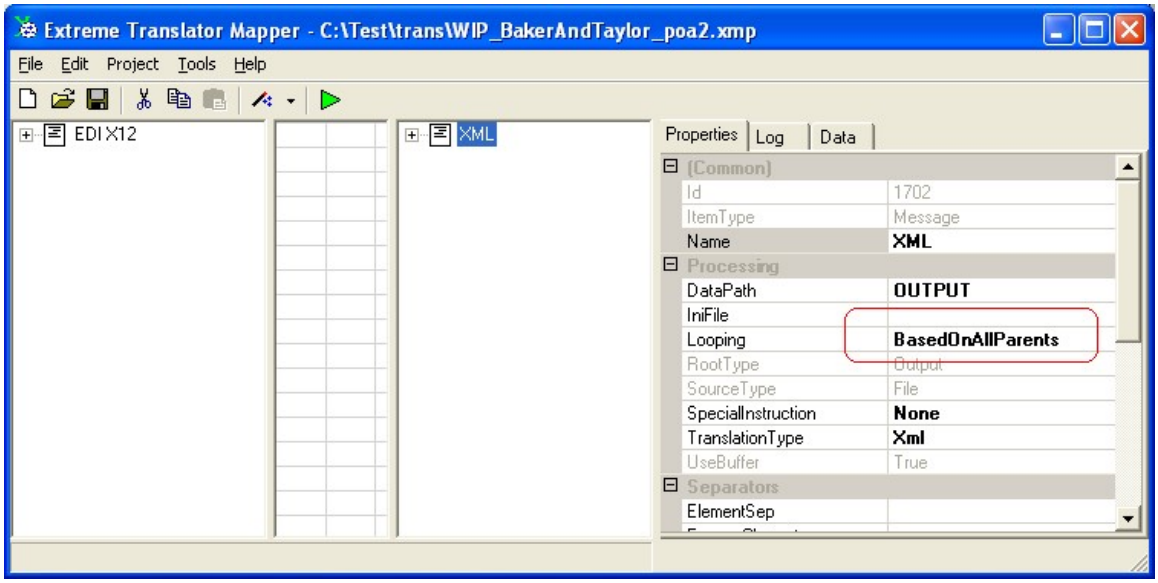
Both “lineitemnumber” and “binding” are in “products” segment, yet some “binding” records are missing when “lineitemnumber” is present all the time and that means “binding” has to be setup

with SpecialInstruction = OutOfLoopData.

2. What are deep nested XML tags? Example:

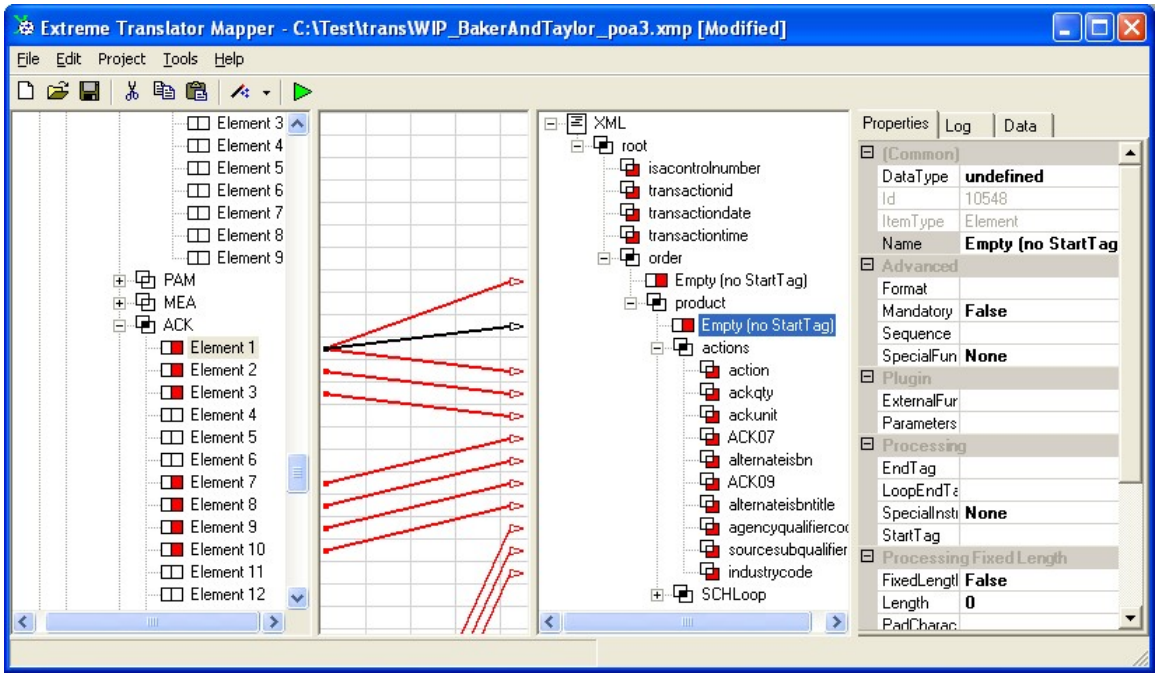
```
<order>
  <orderid>some_id</orderid>  ← Only this is mapped to input
  <product>
    <client>
      <client_data>
        <some_data>  ← Only this part is mapped to some input
      </client_data>
    </client>
 </product>
</order>
```

When XML tags are nested very deep they may have they ordering fixed. The easy fix that works most of the time is changing Looping property on root item of XML tree to BasedOnAllParents.



It may fix deep nested XML tags.

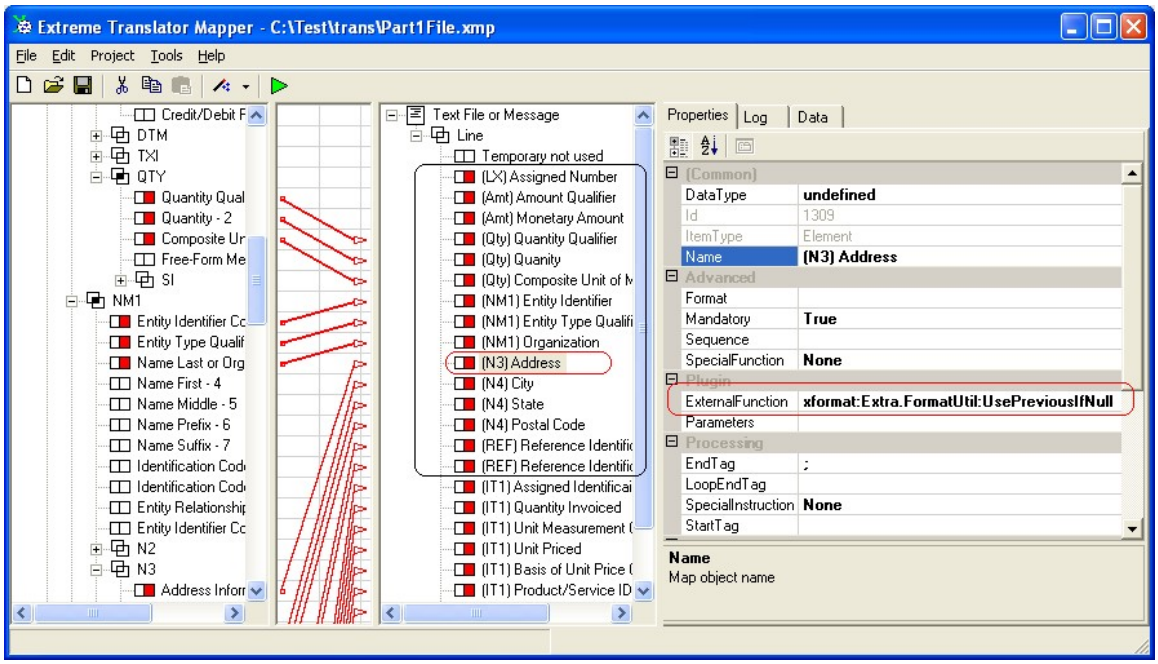
Translator assigns sequence to mapped output items only, so if you have deep nested tags that are not mapped and have nothing mapped in them, then you may still have to go one more step farther in order to fix the looping. That step is simple: add element (attribute) to the segment, do NOT value its StartTag property and map it to item that is also mapped in the child segment. Next picture may explain it better.



Empty elements have been added in deep nested XML to keep XML looping correctly.

Looping issues when mapping X12 or EDIFACT to flat text files

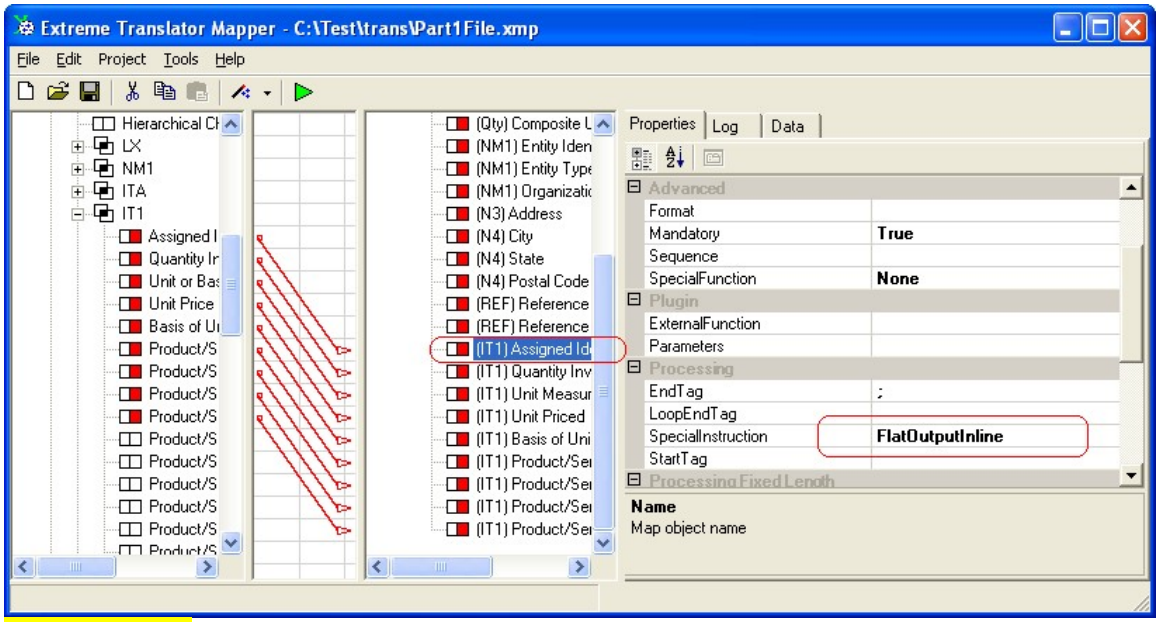
The challenge here is to flatten looping data, and sometimes you may consider producing a few flat files out of X12 or EDIFACT message just because mapping gets too complex when there are 3 or 4 loops in the input and output side has only one line where all it has to go into. If you have something that looks like a header information in the X12 or EDIFACT file and also a one major loop that represents detail lines, then basic mapping is not that complex.



X12 to flat file mapping example. Header section elements are marked in black rectangle. Address information has ExternalFunction setup to UsePreviousIfNull.

There are only two important aspects:
1. Somehow header information has to repeat per each line of flat file when at the same time detail lines should have new values.
2. Header information should correspond to detail lines. If some information is missing in the header, example: some segments are missing in X12 or EDIFACT message, it should not shift detail lines.

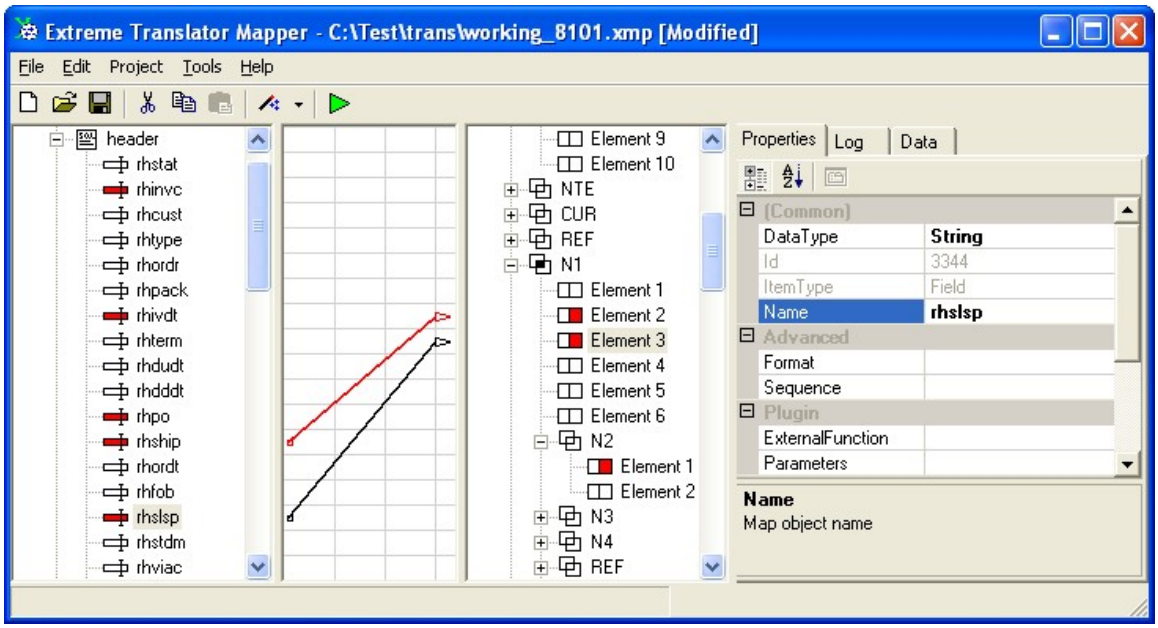
Solutions:
1. In order to keep header information repeating for flat file elements that are mapped to X12 you can set ExternalFunction to UsePreviousValueIfNull on the flat file element. That way it will get repeated.
2. If data is sometimes missing in the X12 or EDIFACT file in the header section, you still can keep it from shifting by setting property SpecialInstruction = FlatOutputInline for the first element on a flat file that comes from detail section. FlatOutputInline is like an instruction for translator that this element is like Primary Key for flat file. Element that always comes in the detail section and is first of all the detail elements.



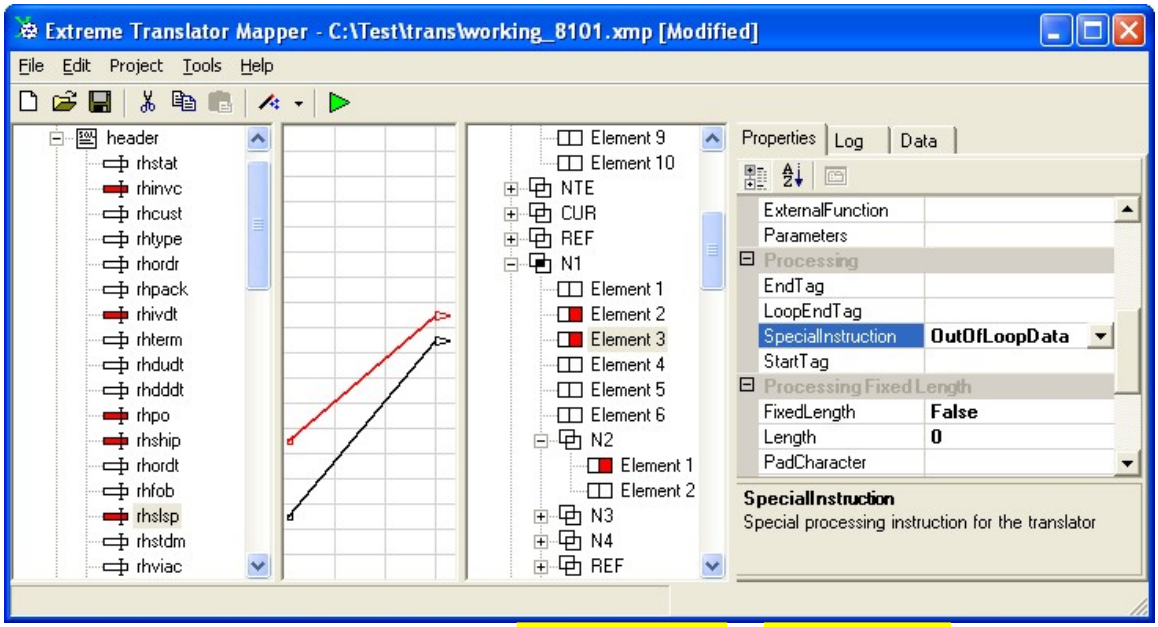
FlatOutputInline usage.

Looping issues when mapping database queries to X12 and EDIFACT

It is best if you could form at least two queries, one to form X12 or EDIFACT header information and one to form detail lines. You can setup relationship between those two queries using map editor (see Database Mappings chapter in general documentation). If you do that then translator will automatically call detail query for each row of header query, and most of map ordering would be handled automatically.
There could be one issue when data in the header on some input fields would be NULL. If segment has nested segments inside of it, and some elements of that segment may contain NULL values, they should be marked with SpecialInstruction = OutOfLoopData to keep nested segments in the right order.



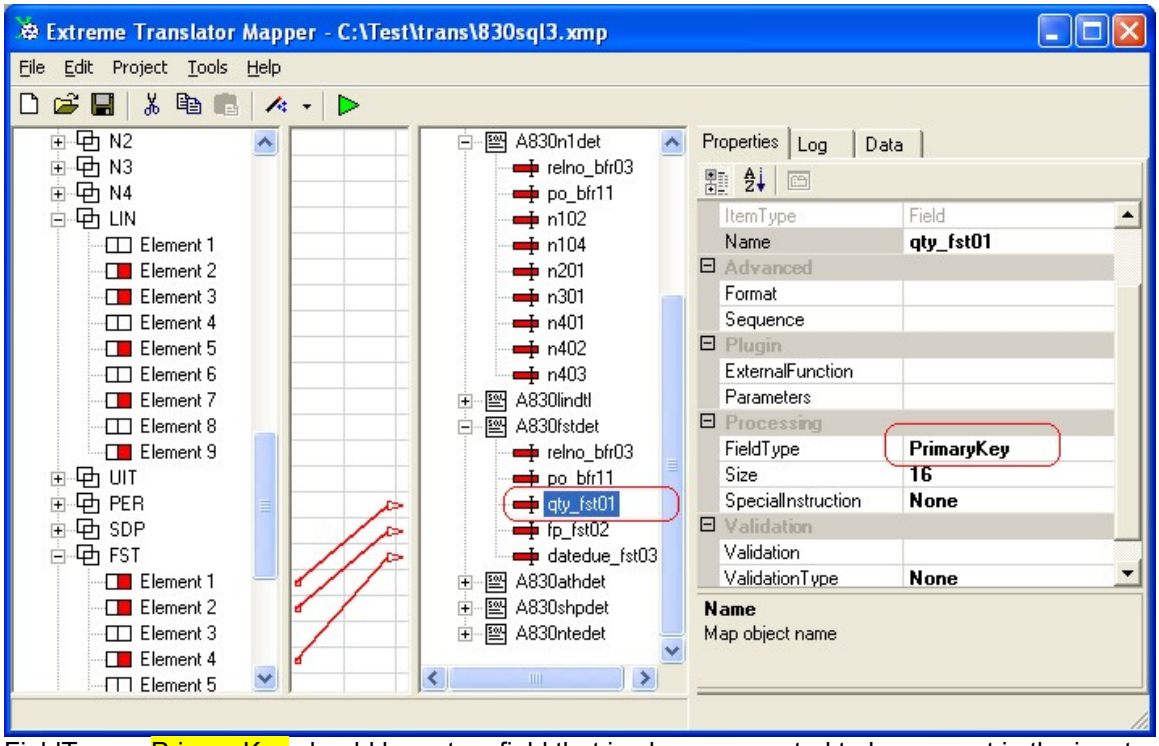
In this example field “rhslsp” might have NULL value in some cases and because of that N2 would be ordered incorrectly in the output.



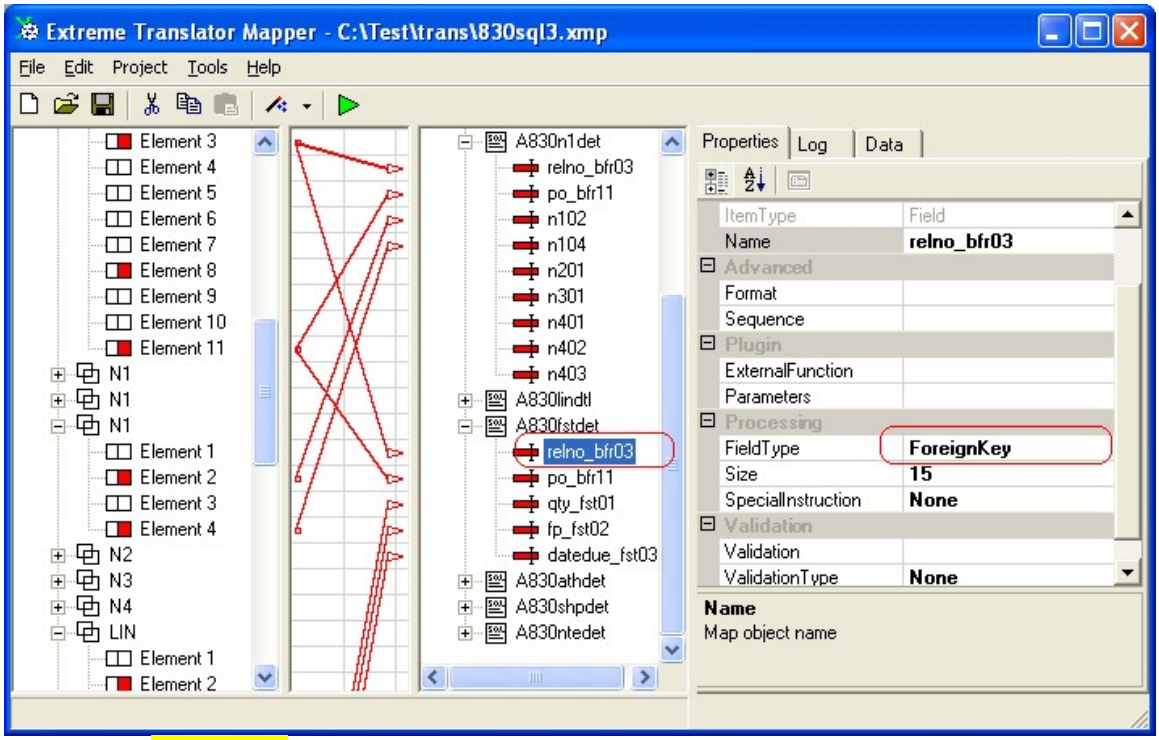
In order to fix this you can change property SpecialInstruction to OutOfLoopData for Element 3.

Row order issues when mapping data to relational databases

Most important property for data mappings to databases is FieldType property. This property translator uses to order records in the database tables. FieldType matches relationship types used in databases, like primary key, foreign key, etc. However actual relationships may not be present in the database for map to work. In other words, those relationships maybe setup only in the map in order to populate tables in desired fashion.



FieldType = PrimaryKey should be set on field that is always expected to be present in the input and have values per each line (record).



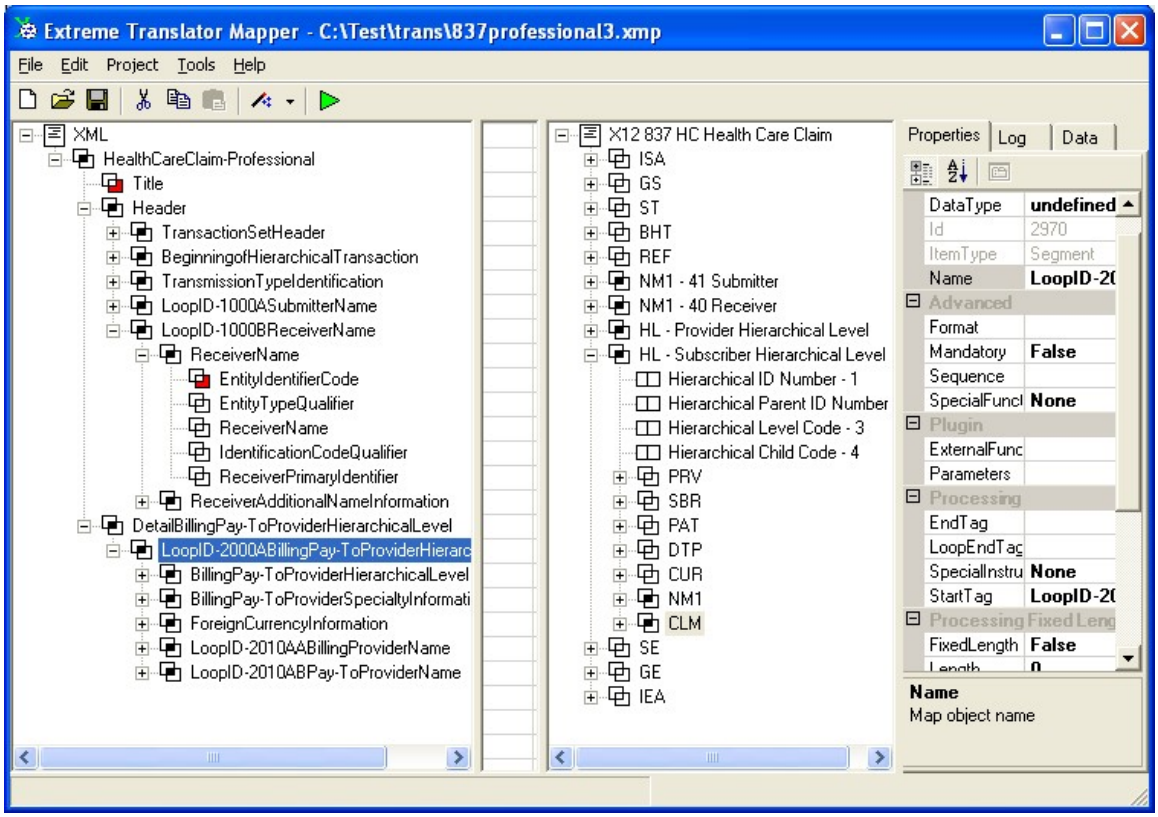
FieldType = ForeignKey should be set on field that comes from some elements above in the input.

In cases when you want to keep same data repeating from one row to the other, Field should be set to FieldType = NotNull, as it will hint translator to copy data from previous record if for this

record field has no value.

Looping issues when mapping XML to X12 or EDIFACT

Most of the time translator will resolve looping automatically if your input is nested same way output is nested. Next picture may explain it better.

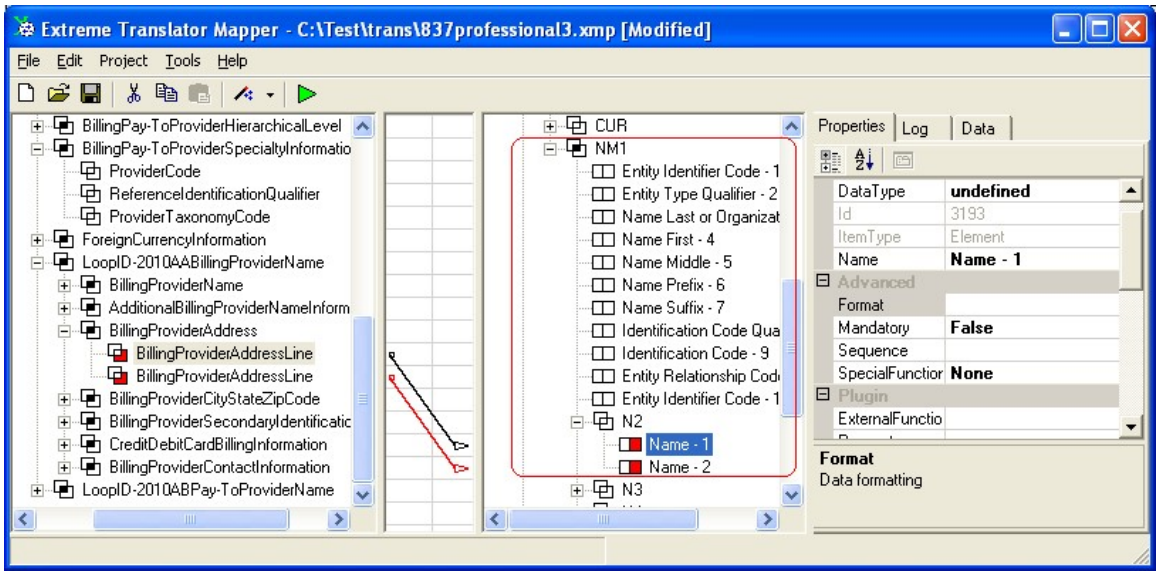


Nested loops on XML side are represented as nested loops on EDI side.

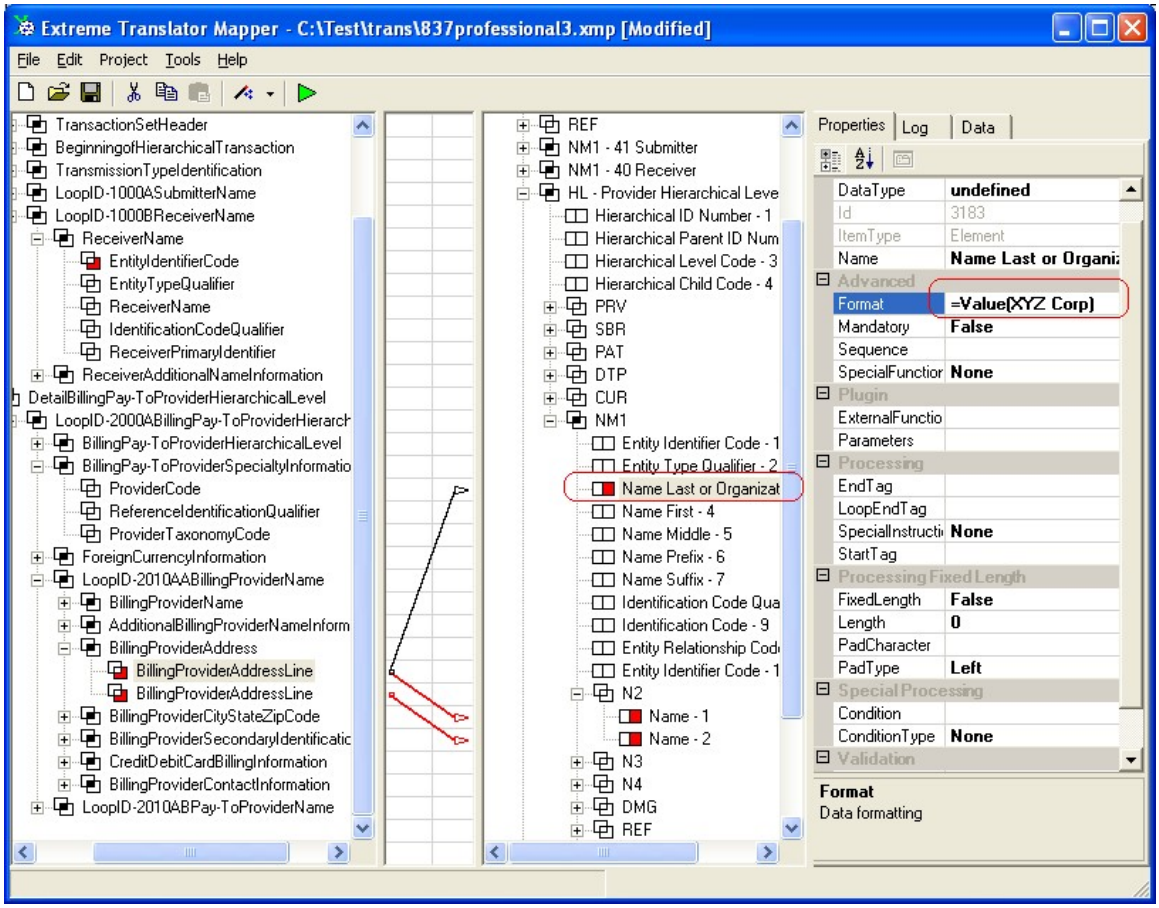
If output EDI segments would not be nested on the left, then translator would not know what is the parent and what is a child of the hierarchy. The nesting of output segments tells translator what has to follow what in looping sequence. That's why for example "HL – Subscriber" holds segments PRV, SBR, etc. until CLM, because they are unique per each repeat of "HL – Subscriber".

However there are cases when loop can go out of sync and for example you would get a few CLMs under one HL. Then that happens fix has to be made to elements that parent loop holds. In this case four elements that are in HL: Hierarchical ID, Hierarchical Parent ID, Hierarchical Level Code, Hierarchical Child Code.

These are different possible cases:
1. None of the elements are mapped (like in the picture). One of those elements can be mapped to some input element that is also used inside the "HL – Subscriber" loop. This extra mapping would hint translator on how data should be ordered in the output.
Next screen shot below has N2 elements mapped but NM1 is not mapped to anything. It is best to map NM1 to the input too.

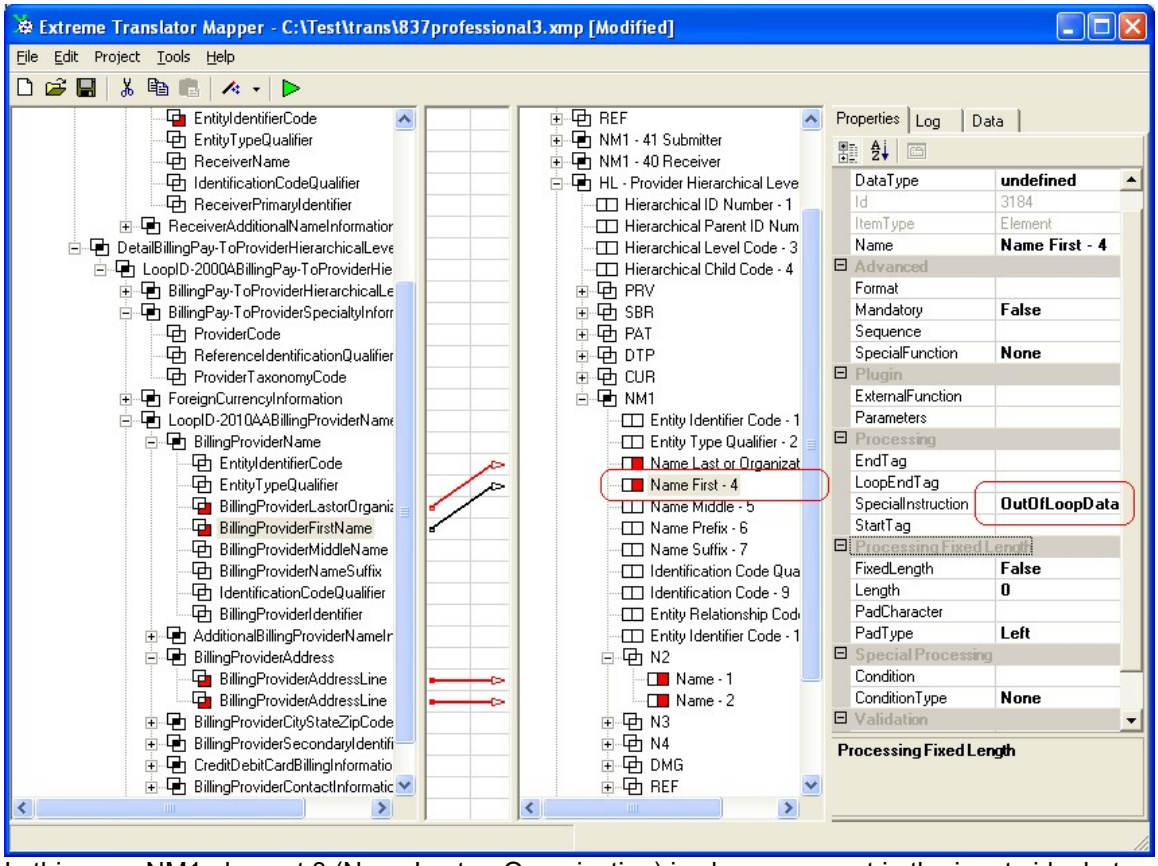


N2 may get out of sequence. Next screen shot is how to fix that.



It is fixed by double mapping input element to parent loop (NM1) and setting constant value in Format property with =Value(XYZ Corp).

2. Next case is when you have no data in the input for one of the parent elements. Example could be same as above, but with two NM1 elements mapped.



In this case NM1 element 3 (NameLast or Organization) is always present in the input side, but NM1 element 4 (Name First) may or may not be there. Missing data can affect looping of child segments, such as N2, N3, etc.

Name First can be marked as SpecialInstruction = OutOfLoopData. Now if it is missing on the input side it will not affect looping of N2 segment.