Section 5: Shipto, depositor, Shipfrom loops, exception, string-join, substring-after, auto-number, replace

Moving on down inside the N1 I will connect a constant of “91” to F66 which means I will be using the “sellers number” and the ‘number’ we will connect will come from the Shipto Adage Name to F67.

Ok moving on to some simple connections. Right click the N3 inside of Loop0100 and duplicate input twice.

Shipto ADDR1 --- > F166 (inside N3)
Shipto ADDR2 --- > F166 (inside N3(2))
Shipto ADDR3 --- > F166 (inside N3(3))

Also notice that if this data is not available Mapforce is not going to create empty segments, its smart enough to leave them out.

Shipto City	--- > N4, F19
Shipto State	--- > N4, F156
Shipto Postal Code	--- > N4, F116

And congratulations you are done with the ST loop0100

No rest for the wicked, on to the next loop. Loop0100 (2) will be our DE (depositor) loop

So create a constant of DE --- > F98_1
Create a constant of your companies name in my case National Frozen --- > F93
If you are using a DUNS create a constant of 9 --- > F66
And then create a constant of the DUNS and connect the ID code for example 123456 --- > F67

And that’s it for Loop0100 (2)

Ok, Loop0100 (3) will be our SF or Ship From loop.

Create a constant of SF --- > F98_1

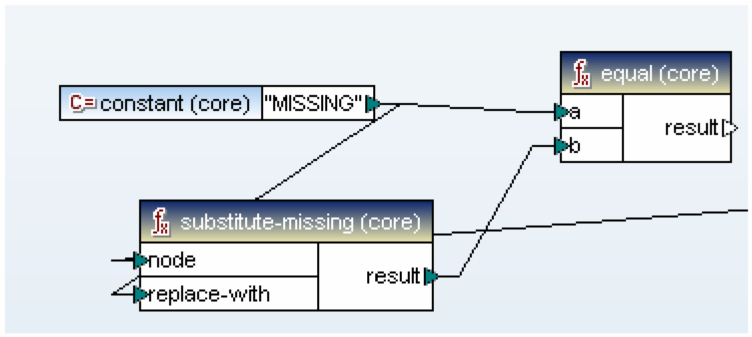
Now for F93 I decided to be extra tricky. I wanted to be sure that there is a value in the name segment for this loop. In order to do so I used an exception. Lets walk through those steps.

First thing to note about using exceptions, it appears you always have to use a filter with one (I could be wrong).

So we need to detect something is missing so drop down the substitute-missing and the connect the SHP FROM WHS to the node.

Now create a constant of the string MISSING and connect it to the replace-with input of the substitute-missing.

Now drag on an equal and connect the missing constant to input a and the result of the substitute-missing to the input b of the equal. It will look something like this.



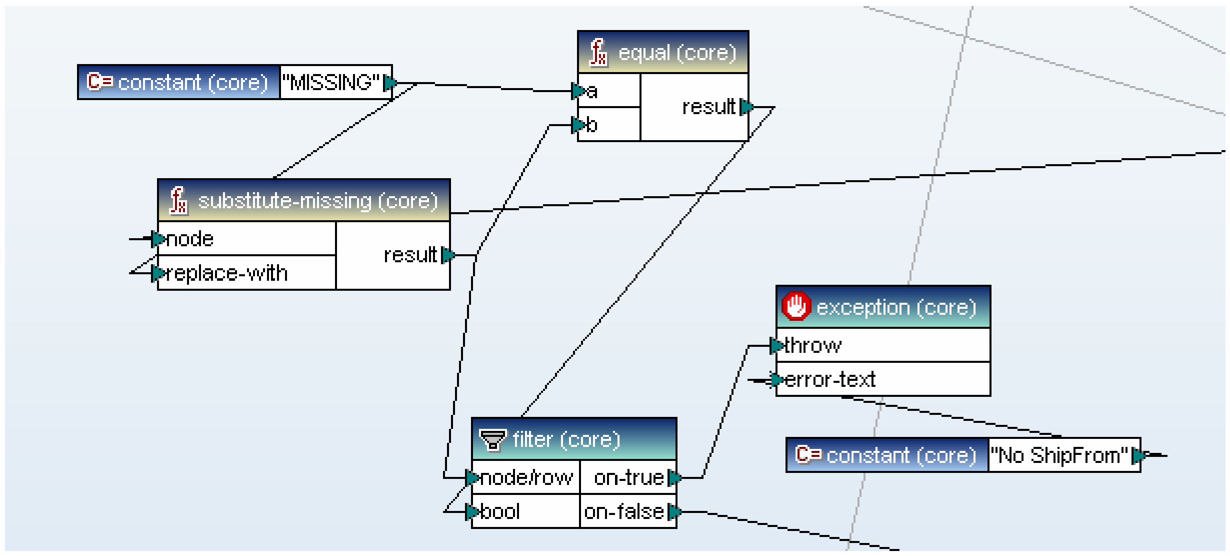
What this gives us is a ‘true’ value if the SHP FROM WHS value is missing.

Now lets create a filter. One quick way is to drag the result of the substitute-missing to F93 and then right click on the line and select insert filter.

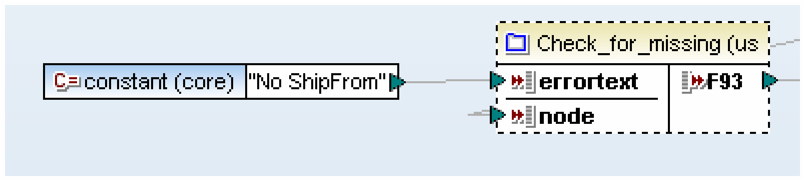
So now the result of the substitute-missing should be connect to the node/row input triangle of the filter, and now we want to connect the result of the equal to the bool input triangle of the filter.

Now if our result was true we want to throw an exception. So lets create an exception. Click the stop sign icon with a hand in the middle now drag from the on-true output from the filter to the throw input on the exception. Now create a constant of the string that you want it to display, in my case “No ShipFrom”.

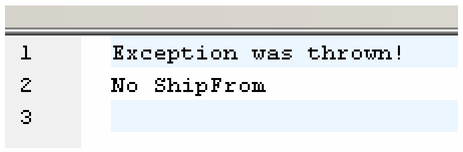
Now make sure the on-false connects to F93. It will look something like this.



You could select all of this and create a user-defined function from selected (leave the value for the error-text out of the selection) and name the function check for missing. Then you could use if for other values as well.



You can test this by opening up the FLATFILE.IN and deleting the PRFX_EXT_PARTNER line, then when you click the output tab it should say.



```
Exception was thrown!
No ShipFrom
```

I’m using a DUNS so now I create a constant of 9 and connect it to F66
Then connect DUNS --- > F67

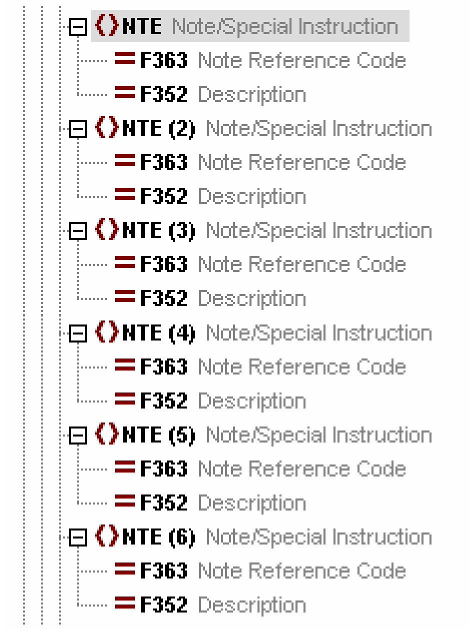
And we are done with Loop0100(3)

We already handled the first G62 in section 1 of the guide. The second G62 is exactly like it, except the F432 Date Qualifier will be a “10” instead of a “2”.

Now we need to grab the Notes for this 940 at a header level.

This part feels a little inelegant but was a quick way for me to do it.

I right clicked on the NTE under my G62 (2) and went duplicate input. I did this five times so it looks like this.

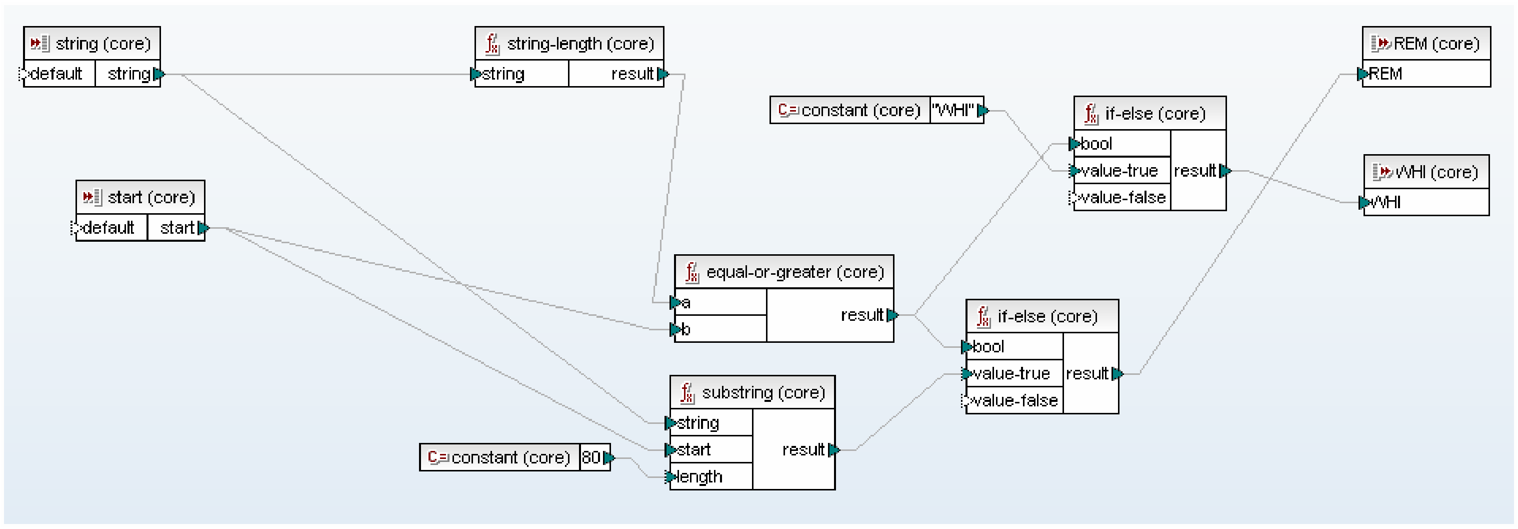


```
NTE Note/Special Instruction
    F363 Note Reference Code
    F352 Description
NTE (2) Note/Special Instruction
    F363 Note Reference Code
    F352 Description
NTE (3) Note/Special Instruction
    F363 Note Reference Code
    F352 Description
NTE (4) Note/Special Instruction
    F363 Note Reference Code
    F352 Description
NTE (5) Note/Special Instruction
    F363 Note Reference Code
    F352 Description
NTE (6) Note/Special Instruction
    F363 Note Reference Code
    F352 Description
```

So the F352 can only hold up to 80 characters so if I have any notes that are longer than this I will split them out, which is why created all these duplicate inputs. I also only want it to fill in a value for the F363 if the description contains data.

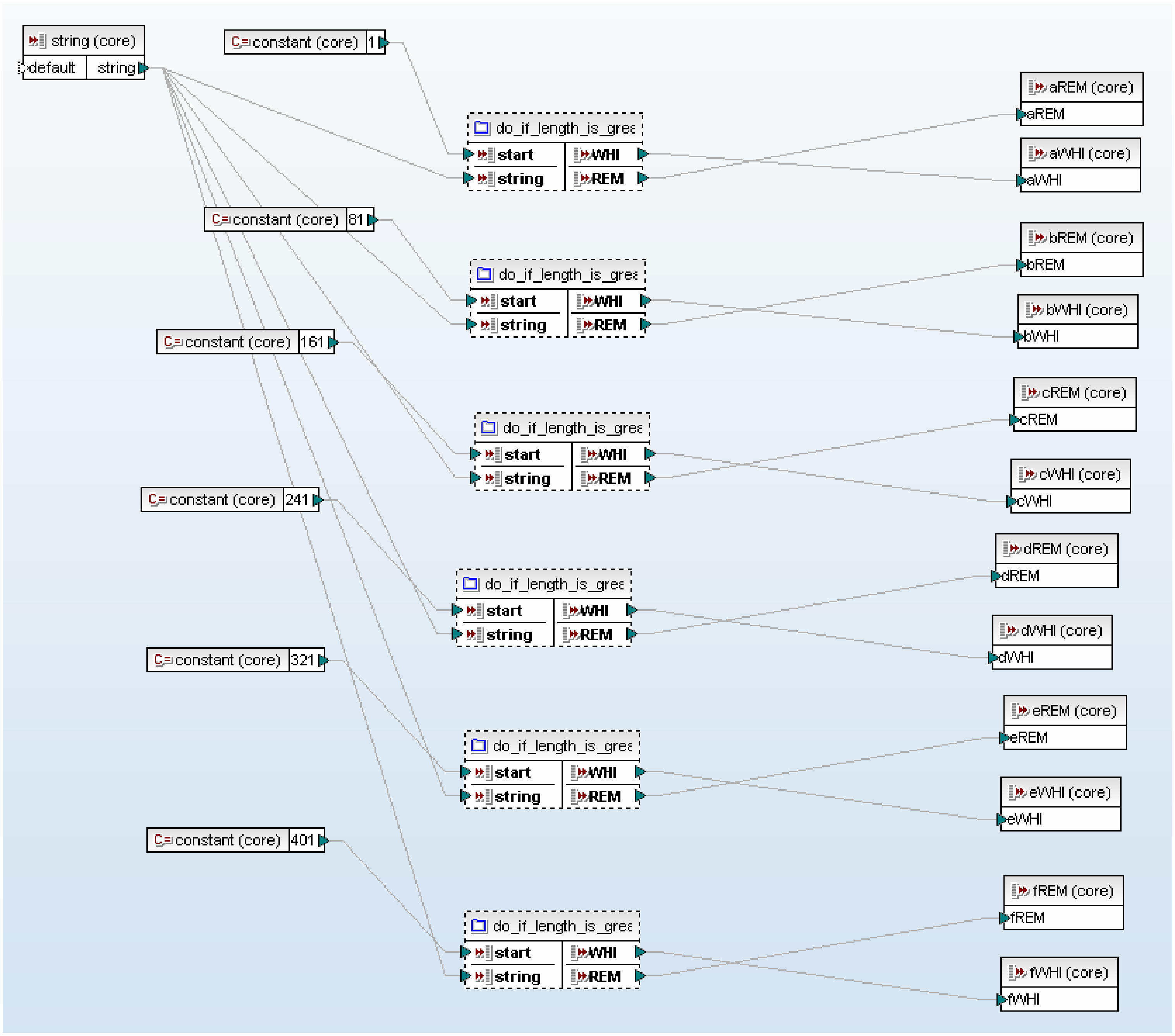
So I created a few functions to achieve this. The innermost function takes a string and then checks its length to see if it is greater than a given value. If it is then it substrings it for 80 chars (or to the end of the string if it is less than 80) and passes out the string to an output of REM and also passes out a string of "WHI" to an output called WHI.

I named this function do_if_length_is_greater, not the greatest name. But it looks like this

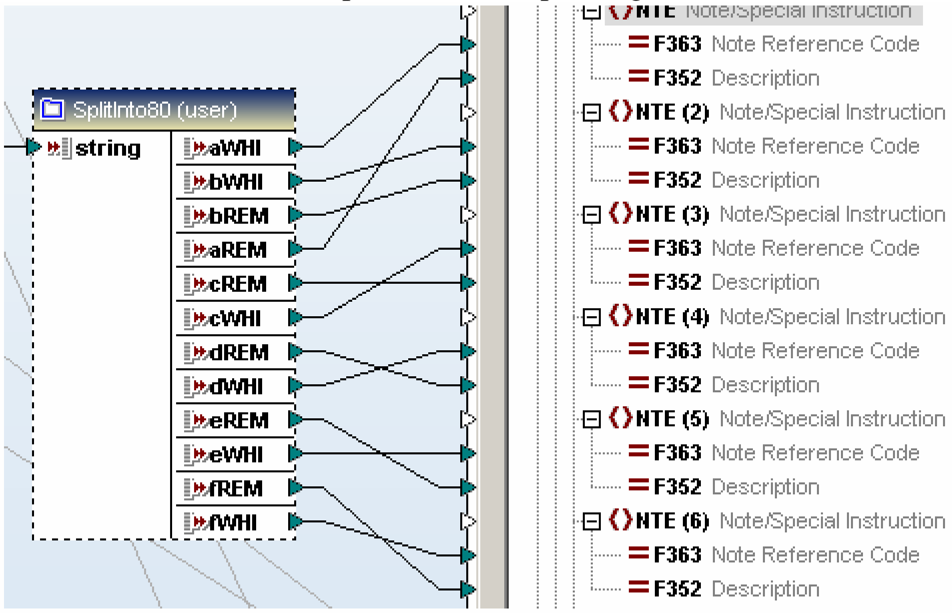


Next I used this function six times. And I created six REM outputs (name aREM through fREM) and six WHI outputs (aWHI through fWHI)

Then I created constant values of 1,81,161,241,321,401 and connect it to 'start' of the do_if_length_is_greater function. It's a monster that I turned into a function named, SplitInto80.



Now I connected each output to its corresponding F363 and F352



Now that we have a string splitter we need to get a string that needs splitting.

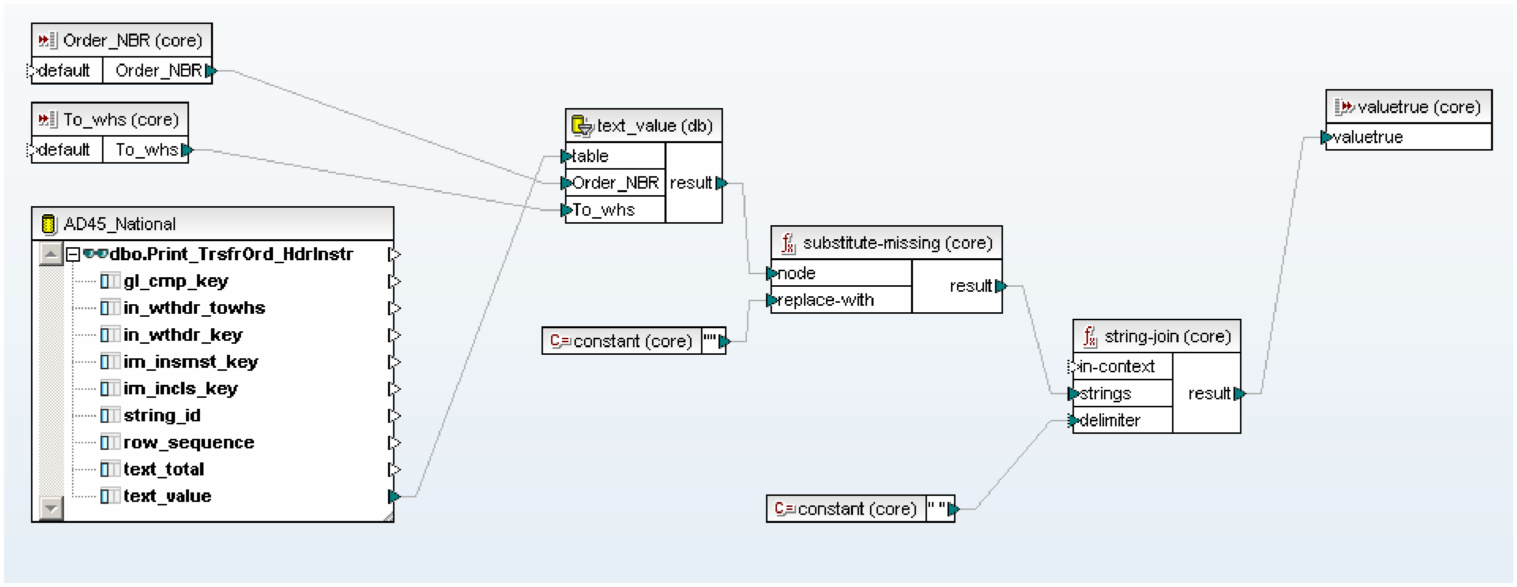
We also need to know if we need to get SO comments or WT comments. This is where we can make use of the isWarehouseTransfer Bool value that is output from the Get_shipto_name_from_adage function we created a while ago.

So I created an if-else and connected the isWarehouseTransfer bool to its bool input triangle.

The Header Instructions (remarks) live in a view another coworker created for her crystal reports.
Since it is possible that there might be multiple comments returned from the one sql-where I put a string-join with a delimiter of " " to handle that possibility.

The SQL-WHERE contained

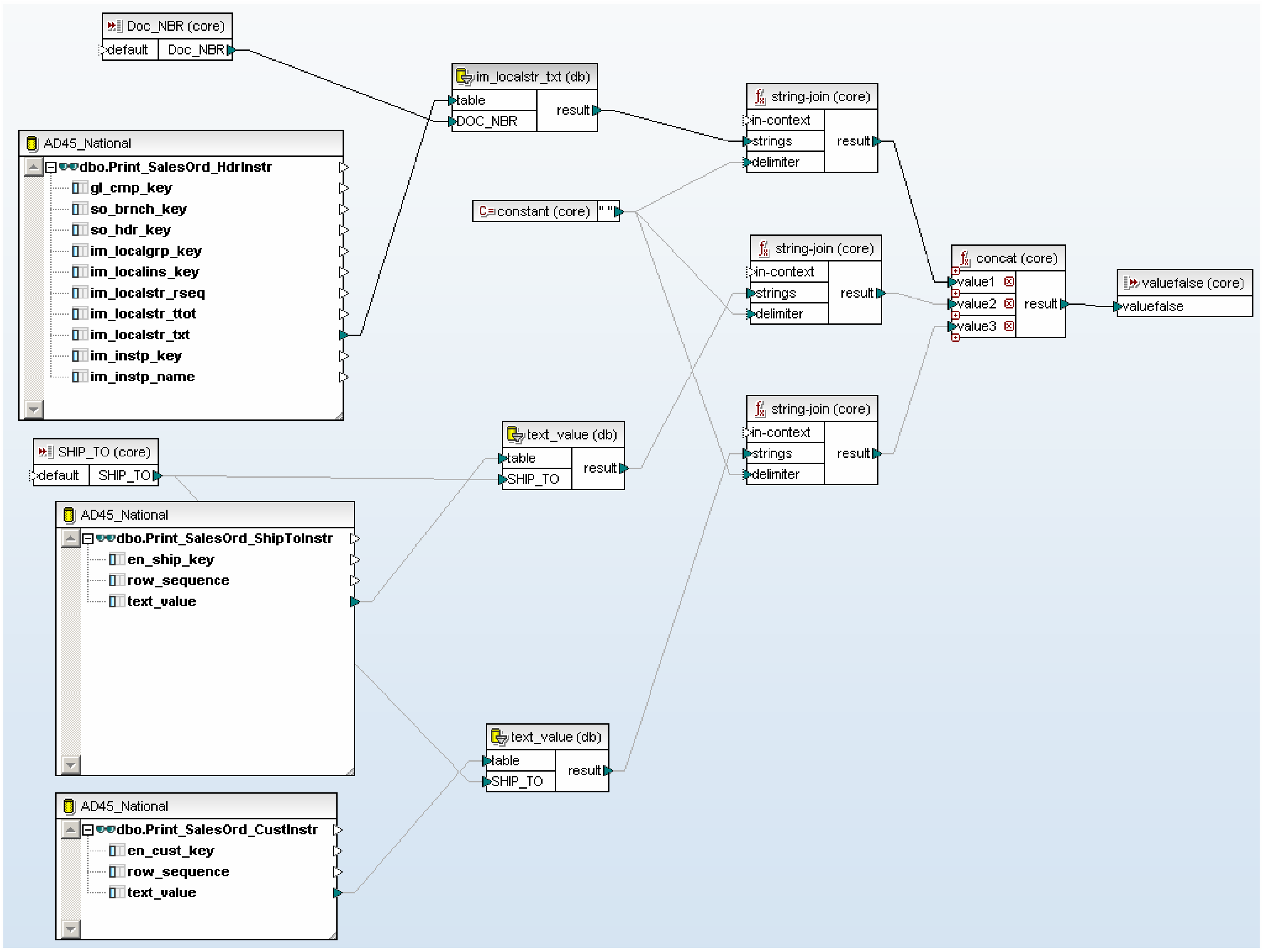
in_wthdr_key = :Order_NBR and in_wthdr_towhs = :To_whs



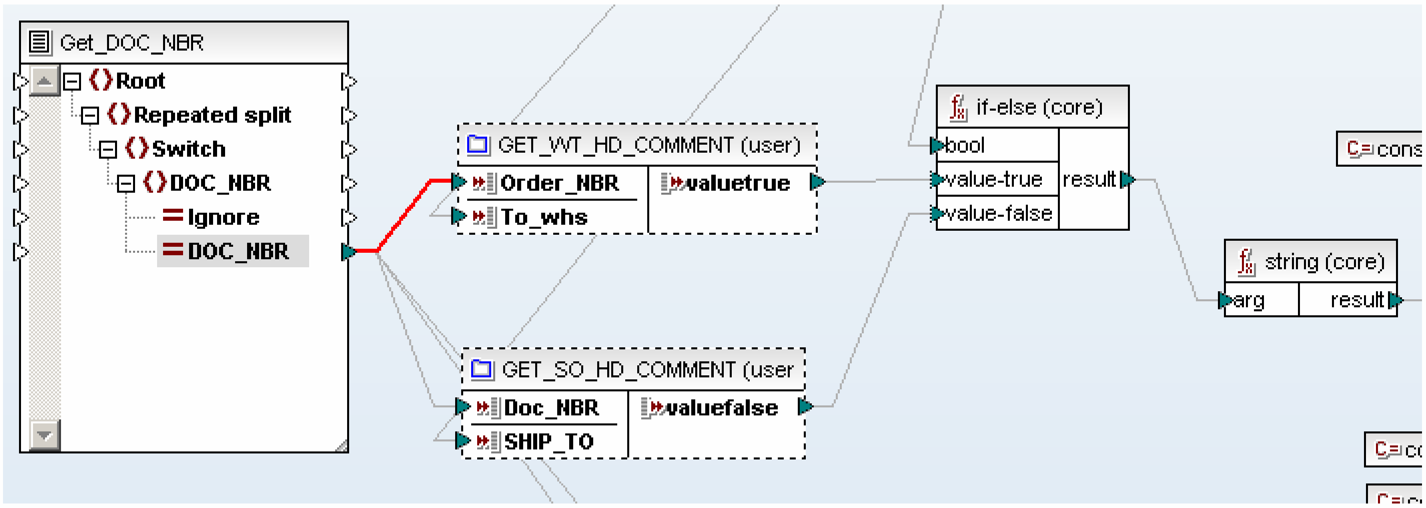
I named this function GET_WT_HD_COMMENT and connected it to the value true of the if-else.

Now I needed to handle the Header comments of a SO. This one was a lot more complicated since the comments could be in any of three different views.

So added the three view and ran three SQL-WHERE's and used three string-joins, and then used a concat to combine them and then connect it to the valuefalse. I named this function GET_SO_HD_COMMENT and then connected it to the value-false of the if-else.



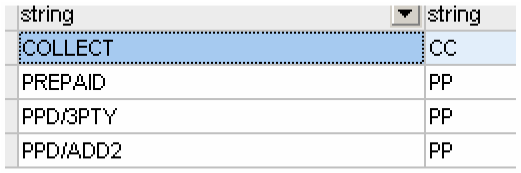
I also needed to supply these functions with the DOC_NBR at the same hierarchical level as the function. If I tried to grab it from the first (main) flex-text it would have instead looped through the function four times cycling through the input variables and never giving the result. So I created a second flex-text that would grab the DOC_NBR and use that to supply GET_WT_HD_COMMENT and GET_SO_HD_COMMENT with the needed value. The end result looks like this.



Ok, now the header section of the notes is done and now we continue on to the W66.

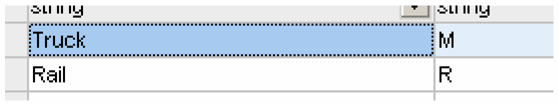
So lets take FREIGHT_CODE apply a value-map and then connect it to F146.

Here's the map



string	string
COLLECT	CC
PREPAID	PP
PPD/3PTY	PP
PPD/ADD2	PP

Do a similar thing with TRANSPORT MODE applying a value-map and then connecting it to F91



string	string
Truck	M
Rail	R

And connect CARRIER SCAC --- > F140

Now we will move into the line detail area of the 940.

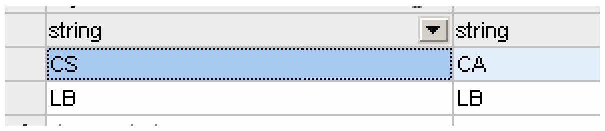
In order to get looping you need to connect some looping source to some looping input. In our case we will connect LN info to Loop0300.

Since our EIF file from Adage does not include line numbers we will have to generate them. So drag on an auto-number. Create a constant of 1 and then connect it to start_at, and a constant of 1 connected to increase. Then connect the result of the auto-number to F554 (under LX) Any time it loops it will increment and created the Assigned number segment.

Now we skip over everything until will get to Loop0310, W01 line Item detail, F380 Order quantity.

We want to connect the ORD QTY to F380 but if you remember in order to get all of this line at the same 'level' I used a save as CSV. What this means is the value in ORD QTY contains the string "DT_ORD_QTY=" and then the value. So in order to get rid of the string I used a substring-after. I connected ORD QTY to the string input of the substring-after and created a constant string of "DT_ORD_QTY=" and connect it to the substr input. I connect the result of the substring to a 'number' so that it will convert it before connecting it to the F380.

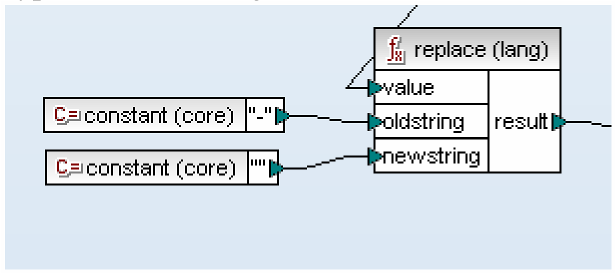
Next is F355 which gets its value from ORD UOM which also passes through a substring after that removes "DT_ORD_QTY_UOM=" and then the result of that through a value-map.



string	string
CS	CA
LB	LB

For F235_1 I created a constant of "VN" and connected it.

Next is the Product/Service ID. This is stored with a hyphen and we will need to remove it. First connect the ITM ID to a substring-after that removes "DT_ITM_ID=" then the result of that will go to a replace. Create a constant of hyphen "-" and connect it to oldstring, and then a string of "" and connect it to newstring. This will replace the hyphen with nothing. And then connect the result to F234_1



For F369 (under G69) connect PROD DESC to a substring-after and remove "DT_PROD_DESC="

Now we move on to the line level detail comments.

Once again I right clicked and created six NTE nodes.

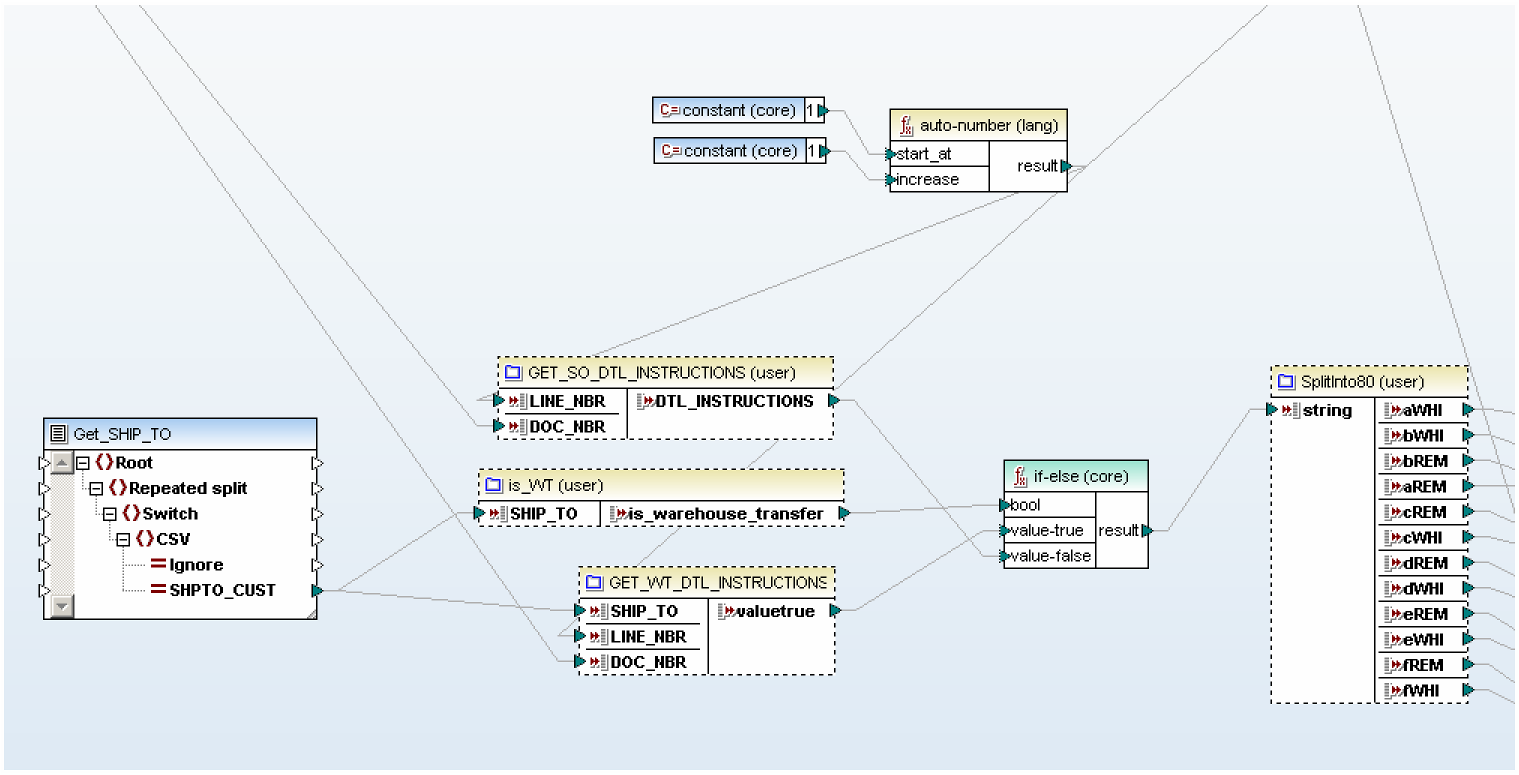
Then I reused my SplitInto80 function and connected the WHI and the REM outputs.

I created another function to test whether this was a SO or WT. I think I could have connected the previously used bool but I didn't want to confuse any code generation.

Also to simply things for me I created another flex-text and used to it grab the SHPTO_CUST for my database queries.

Just like before I created an if-else. I created two more functions, GET_SO_DTL_INSTRUCTIONS and GET_WT_DTL_INSTRUCTIONS

The only difference with these functions is that they take a LINE_NBR as an input for the SQL-WHERE query. It also uses a string-join in case of multi line returns from the query.



Now the final piece is the F380 (under W76)

In order to get the total of the line I use a sum. I get the values from the ORD QTY (which passes through a substring-after) and for the in-context I used the root of my main flex-text.

I then compile this code and wrote some batch jobs to import the Data into TrustedLink to send off.