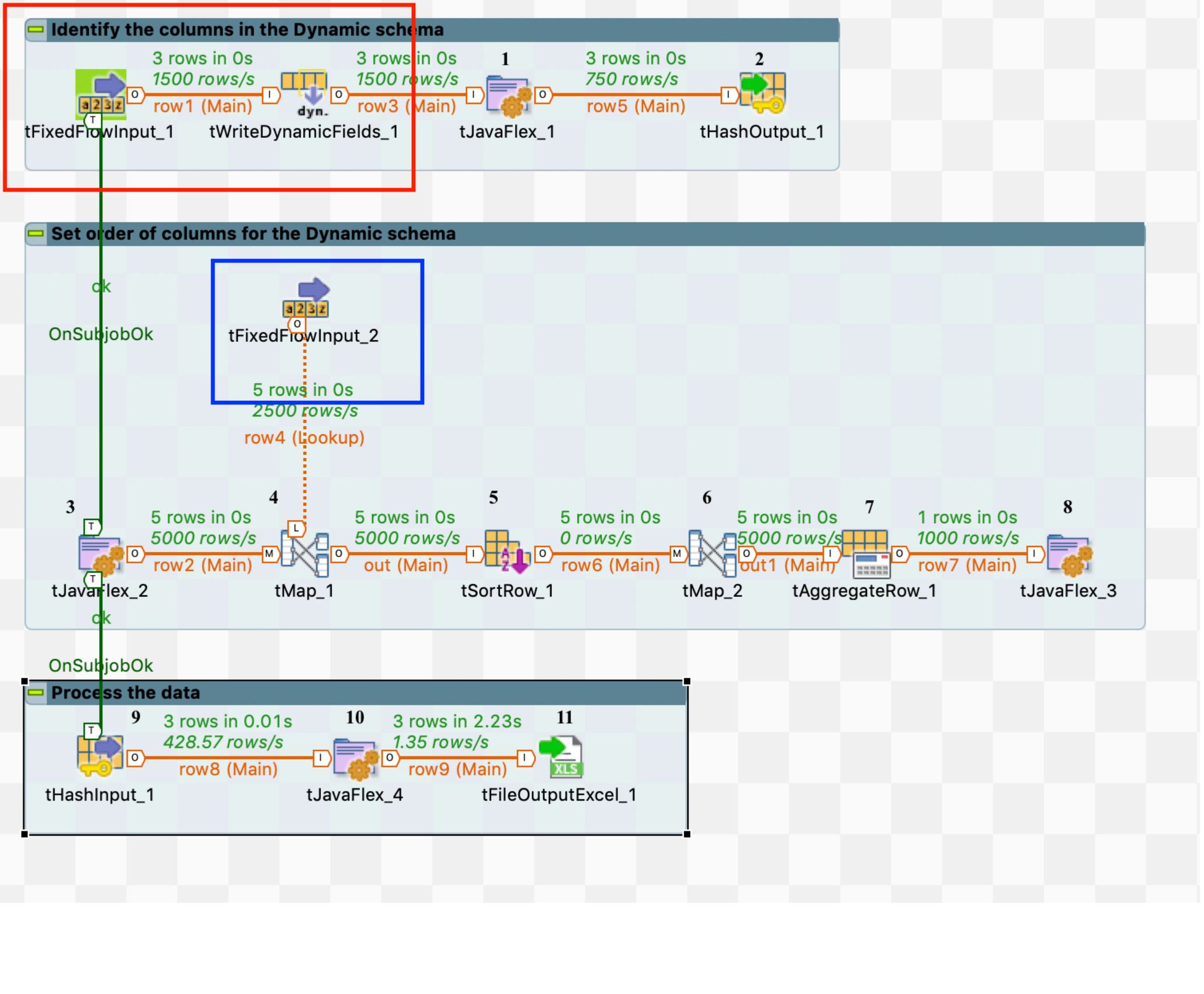
Sorry about the delay in getting back to you. I had to figure out a good way of doing this. I *might* even write a blog on it, but I figured it would be mean to keep you waiting for that.

The fact that you are using the subscription version makes this possible. I'll demonstrate a job which does this below. You will have to modify it to accommodate your data sources.

The job looks like this.....


Identify the columns in the Dynamic schema
3 rows in 0s
1500 rows/s
row1 (Main)
tFixedFlowInput_1
3 rows in 0s
1500 rows/s
row3 (Main)
tWriteDynamicFields_1
1
3 rows in 0s
750 rows/s
row5 (Main)
tJavaFlex_1
2
tHashOutput_1
Set order of columns for the Dynamic schema
ok
OnSubjobOk
tFixedFlowInput_2
5 rows in 0s
2500 rows/s
row4 (Lookup)
3
5 rows in 0s
5000 rows/s
row2 (Main)
tJavaFlex_2
4
tMap_1
5
5 rows in 0s
5000 rows/s
out (Main)
tSortRow_1
6
5 rows in 0s
0 rows/s
row6 (Main)
tMap_2
7
5 rows in 0s
5000 rows/s
out1 (Main)
tAggregateRow_1
8
1 rows in 0s
1000 rows/s
row7 (Main)
tJavaFlex_3
ok
OnSubjobOk
Process the data
9
3 rows in 0.01s
428.57 rows/s
row8 (Main)
tHashInput_1
10
tJavaFlex_4
3 rows in 2.23s
1.35 rows/s
row9 (Main)
11
tFileOutputExcel_1

The components surrounded in the red box are there to produce some test data. You will need your data source there and it must use a single Dynamic schema column. The component surrounded by the blue box is the data source which holds the column renaming/ordering data. This will need to be replaced by your data source for this.

I will go on to describe each of the components in order.

1 - tJavaFlex_1

This component is used to dig into the Dynamic schema and extract the column names and order of the columns in your data source. The code for this is shown below.....

Start Code

```
//Create an ArrayList to contain the column names of the input source
java.util.ArrayList<String> columns = new java.util.ArrayList<String>();
//Row count variable to count rows processed
int rowCount = 0;
```

Main Code

```
//Only carry out this code for the first row
if(rowCount==0){
 //Set the dynamicColumnsTmp variable
 Dynamic dynamicColumnsTmp = row3.dynamicColumns;

 //Cycle through the columns stored in the Dynamic schema column and add the column names
 //to the ArrayList
 for (int i = 0; i < dynamicColumnsTmp.getColumnCount(); i++) {
     DynamicMetadata columnMetadata = dynamicColumnsTmp.getColumnMetadata(i);
       columns.add(columnMetadata.getName());

 }

 //Append 1 to the rowCount
 rowCount++;
}
```

End Code

```
//Set the columns ArrayList to the globalMap for later
globalMap.put("columns", columns);
```

2 - tHashOutput_1

This component is used to simply store the data that is passed through the tJavaFlex from your source. This will be used later when the data is processed.

3 - tJavaFlex_2

This component is used to return the data collected about columns in the first tJavaFlex and return it one at a time as data rows. The code is described below....

Start Code

```
//Retrieve the columns ArrayList to be used here
java.util.ArrayList<String> columns = (java.util.ArrayList<String>)globalMap.get("columns");
java.util.Iterator<String> it = columns.iterator();

//Set a loop to produce a row for each column name
while(it.hasNext()){
```

Main Code

```
//Set each column name to a new row
row2.columnName = it.next();
```

End Code

```
// end of the component, outside/closing the loop
}
```

4 - tMap_1

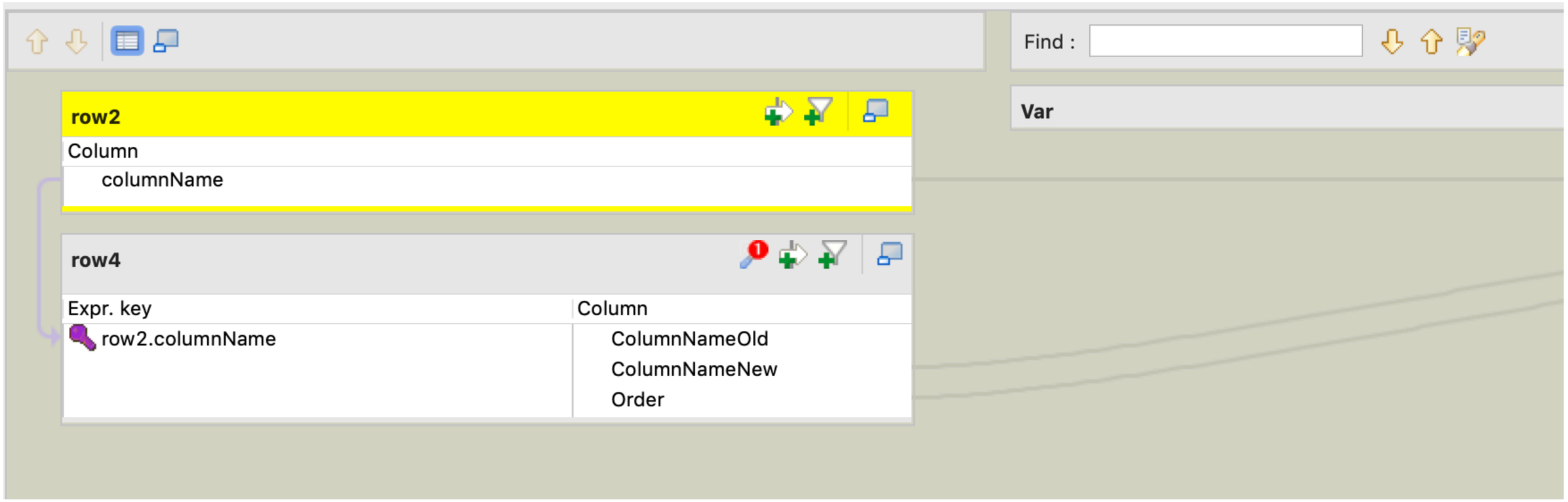
This component is used to join the column data derived from the first subjob with the column translation data provided by your translation source. The schema I am using for the column translation source is....

ColumnNameOld - String

ColumnNameNew - String

Order - Integer

We will join the ColumnNameOld to the columns derived from the first subjob, The tMap is shown below...



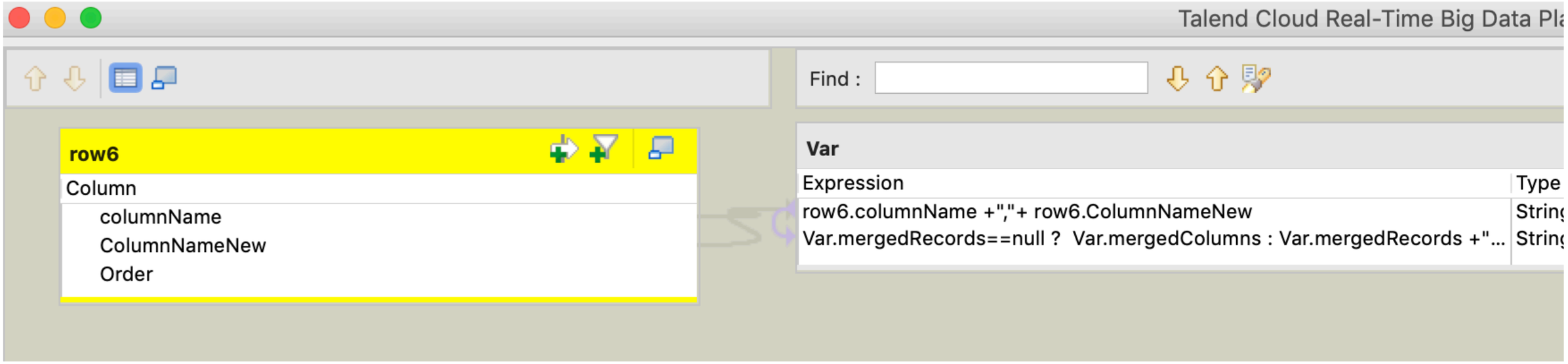
We output the column data derived in the first subjob linked to the new column name and the order.

5 - tSortRow_1

This component is used to sort the rows in the order they need to be output. This is using the Order column.

6 - tMap_2

This component is used to create a single String of the old column name with the new column name in order. These are separated by comma (,) for the old column name and then the new column name, then a semi-colon for the column name combinations in order. These are appended using the tMap variables (which store their values between rows). They are then output so that the complete String is returned in the last row. The following component (tAggregateRow) is used to return only the last of these rows.

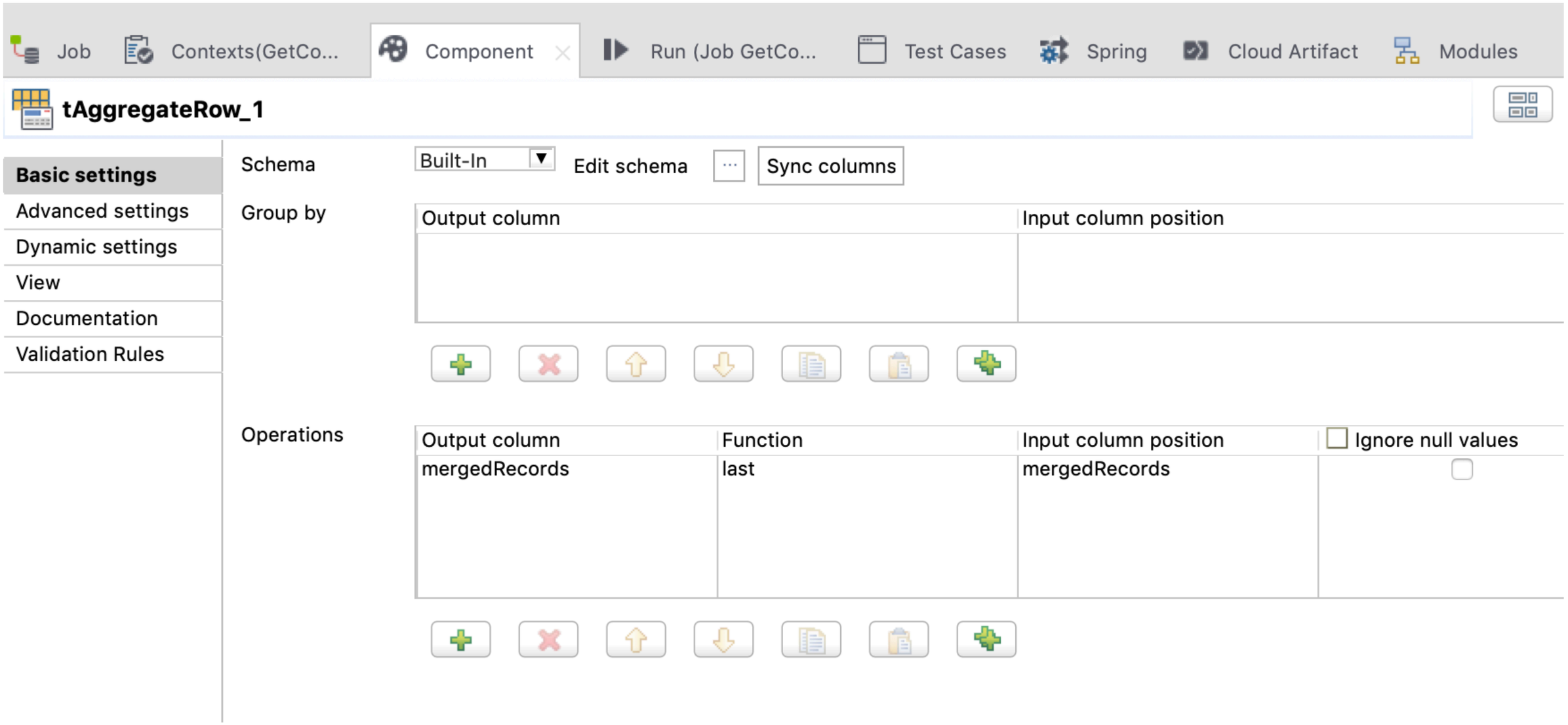


The tMap variable names and values are shown below...

Variable Name	Expression
mergedColumns	row6.columnName +","+ row6.ColumnNameNew
mergedRecords	Var.mergedRecords==null ? Var.mergedColumns : Var.mergedRecords +";"+Var.mergedColumns

7 - tAggregateRow_1

This component is used to reduce the output of the last component to just 1, the final record.



8 - tJavaFlex_3

This component is used to set the returned column order String to a globalMap. I could have used a tSetGlobalVar component for this. The code is shown below....

Main Code

```
//Set the column translation record to the globalMap
globalMap.put("records", row7.mergedRecords);
```

9 - tHashInput_1

This component is used to return the main dataset saved in the tHashOutput in the first subjob. These will need to be linked.

10 - tJavaFlex_4

This component is used to reorder the columns inside the Dynamic schema. This is done in Java and must be followed exactly. The code is shown and described in-line below....

Start Code

```
//Retrieve the "records" globalMap String which holds the record order
String records = ((String)globalMap.get("records"));
//Splite the columns up using the semi-colon
String[] columns = records.split(";");
```

Main Code

```
//Create a Dynamic schema variable to hold the incoming Dynamic column
routines.system.Dynamic dynamicColumnsTmp = row8.dynamicColumns;
//Create a brand new Dynamic column variable to be used for the newly formatted record
routines.system.Dynamic newDynamicColumns = new routines.system.Dynamic();

//Cycle through the column data supplied by the globalMap
for(int x = 0; x<columns.length; x++){
 //Cycle through the columns inside the Dynamic column holding the data
 for (int i = 0; i < dynamicColumnsTmp.getColumnCount(); i++) {
  //Retrieve the value of the current column inside the Dynamic column
    Object obj = dynamicColumnsTmp.getColumnValue(i);
    //Retrieve a DynamicMetadata object from the column inside the Dynamic column
       DynamicMetadata columnMetadata = dynamicColumnsTmp.getColumnMetadata(i);

  //If the current column inside the Dynamic column starts with same name
 if(columns[x].startsWith(columnMetadata.getName()+",")){
   //Identify the old and new column names from the column record
   String newColumnName = columns[x].substring(columns[x].indexOf(',')+1);
   String oldColumnName = columns[x].substring(0,columns[x].indexOf(','));
   //Create a new DynamicMetadata object
   DynamicMetadata tmpColumnMetadata = new DynamicMetadata();
   tmpColumnMetadata.setName(newColumnName);
   //Set the metadata for this metadata
   tmpColumnMetadata.setDbName(columnMetadata.getDbName());
 tmpColumnMetadata.setType(columnMetadata.getType());
 tmpColumnMetadata.setDbType(columnMetadata.getDbType());
 tmpColumnMetadata.setLength(columnMetadata.getLength());
 tmpColumnMetadata.setPrecision(columnMetadata.getPrecision());
 tmpColumnMetadata.setFormat(columnMetadata.getFormat());
 tmpColumnMetadata.setDescription(columnMetadata.getDescription());
 tmpColumnMetadata.setKey(columnMetadata.isKey());
 tmpColumnMetadata.setNullable(columnMetadata.isNullable());
 tmpColumnMetadata.setSourceType(columnMetadata.getSourceType());

 //Set the new metadata for the new column inside the new Dynamic schema column
   newDynamicColumns.metadatas.add(tmpColumnMetadata);

   //Set the value for the new column inside the new Dynamic schema column
   newDynamicColumns.addColumnValue(obj);

  }
}
```

```
}

//Set the output Dynamic schema column
row9.dynamicColumns = newDynamicColumns;
```

Once the data leaves this component, the columns inside the Dynamic schema column will be reordered.

11 - tFileOutputExcel_1

This component writes the data out to your Excel file. Make sure you set the "Include Header" tickbox to true.

If you followed this exactly, you should see that you can dynamically change the order of columns inside your Dynamic schema column by simply changing the data which stores the required column order (Step 4).

I hope this helps.