



ANGEWANDTE
INFORMATIONSTECHNIK
Forschungsgesellschaft mbH



Handbook

Pentaho-Kettle

Version Draft

2009-08-10



Pentaho-Kettle

Project Number			
Project Title	Handbook		
Document Reference Title	Pentaho-Kettle	Date	2009-08-10
Document Name	Pentaho Handbook	Version Draft	draft <input checked="" type="checkbox"/> final <input type="checkbox"/>
Restrictions	public <input type="checkbox"/>	internal <input type="checkbox"/>	restricted <input checked="" type="checkbox"/> :
Distribution			
Authors	Jürgen Pammer		
Abstract			
Keywords			
Document Revisions			
Version	Date	Author(s)	Description of Change



Table of Contents

- 1 Management Summary..... 5**
 - 1.1 Current PHP based version 5
 - 1.2 Pentaho Kettle Data Integration 5
 - 1.3 Comparison between PHP and Pentaho Kettle 5
- 2 Pentaho Kettle Data Integration – User Interface 6**
- 3 Pentaho Kettle Data Integration - Transformations 8**
 - 3.1 Input Steps 8
 - 3.1.1 *Excel Input*..... 8
 - 3.1.2 *Generate Rows* 10
 - 3.1.3 *Get File Names* 11
 - 3.1.4 *Get data from XML* 13
 - 3.1.5 *Table Input*..... 16
 - 3.2 Output Steps 17
 - 3.2.1 *Excel Output*..... 18
 - 3.2.2 *Text file output* 20
 - 3.2.3 *XML Output* 21
 - 3.3 Transformation Steps 24
 - 3.3.1 *Add XML*..... 24
 - 3.3.2 *Add constants* 25
 - 3.3.3 *Calculator*..... 26
 - 3.3.4 *Replace in string* 28
 - 3.3.5 *Row flattener* 28
 - 3.3.6 *Select Values* 28
 - 3.3.7 *Sort Rows* 30
 - 3.3.8 *Split fields* 30
 - 3.3.9 *Unique rows* 32
 - 3.3.10 *XSL Transformation* 32
 - 3.4 Scripting Steps 33
 - 3.4.1 *Modified Java Script Value* 33
 - 3.5 Lookup Steps..... 34
 - 3.5.1 *HTTP client* 34
 - 3.6 Join Steps 35



- 3.6.1 Merge Join..... 35
- 3.7 Job Steps 37
 - 3.7.1 Get Variables..... 37
 - 3.7.2 Set Variables 37
- 4 Pentaho Kettle Data Integration - Jobs 39**
 - 4.1 General entries 39
 - 4.1.1 START 39
 - 4.1.2 DUMMY..... 39
 - 4.1.3 Abort Job 40
 - 4.1.4 Display MsgBox Info 40
 - 4.1.5 Job..... 40
 - 4.1.6 Transformation 42
 - 4.2 Mail entries 42
 - 4.2.1 Mail..... 43
 - 4.3 File management 43
 - 4.3.1 Create folder 43
 - 4.3.2 Create file 44
 - 4.3.3 Delete file..... 44
 - 4.3.4 Delete files 44
 - 4.3.5 Delete folders 45
 - 4.3.6 Move files..... 45
 - 4.4 Conditions..... 45
 - 4.4.1 Simple Evaluation..... 46
 - 4.5 XML..... 46
 - 4.5.1 47
 - 4.6 File Transfer 47
 - 4.6.1 Get File with FTP..... 47
 - 4.6.2 Put a file with FTP 48
 - 4.7 Simple example how to combine some job entries and transformations 49
- 5 Practical Examples 52**
 - 5.1 ISPAN Data (DISMARC) 52
 - 5.1.1 Pentaho Data Integration Transformation..... 55
 - 5.2 BNF Data (BHL) 70
- Glossar..... 74**
- I. Liste verbundener Dokumente 76**
- II. List of Figures 77**



III. List of Tables..... 84



1 Management Summary

This document illustrates the use of Pentaho Kettle Data Integration with examples of the DISMARC project. Similarly to the PHP mappers, Kettle is used to import data of different formats (Excel sheets, databases), transform and export it again via FTP. An overview of the current PHP based version is included too. A summary of Kettle DI is given and various transformation steps are explained as well as two example tutorials including the ISPAN and WOMAX mappings.

Another section compares the use of PHP with Kettle and points out advantages and disadvantages of both ways of data integration.

1.1 Current PHP based version

1.2 Pentaho Kettle Data Integration

Pentaho Data Integration is an ETL¹ tool with a graphical user interface, which allows the user to create transformations and jobs by Drag and Drop. It offers many transformation steps and job entries that can be used to create the desired output. In Kettle Transformations and Jobs can be created. In a Transformation various transformation steps can be applied to an imported data (e.g. tables from an Excel sheet). A job on the other hand executes various actions called job entries, such as getting data from an FTP server or putting it back on it. It can call transformations or even other jobs.

These functions are explained in the further sections of the document.

1.3 Comparison between PHP and Pentaho Kettle

¹ ETL ... Extract, Transform, Load

2 Pentaho Kettle Data Integration – User Interface

After creating a new transformation by clicking File->New->Transformation, or alternatively CTRL+N the transformation step categories can be chosen on the left side of the user interface when choosing Design.

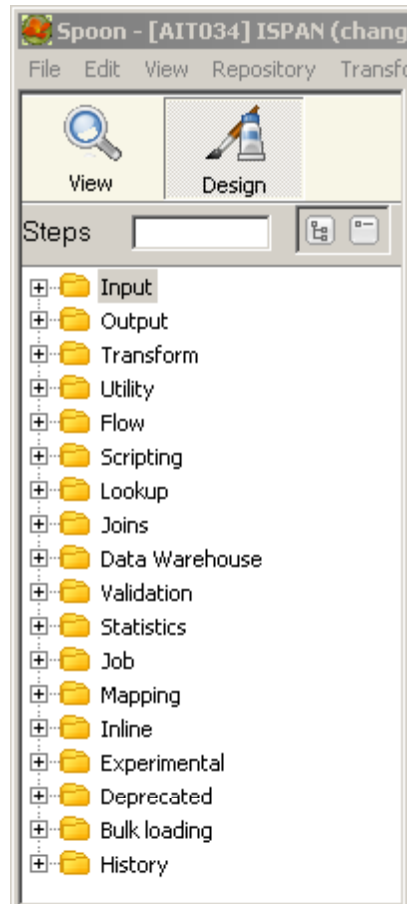


Figure 2-1 Categories of transformation steps

The different symbols can be dragged to the workspace and by double-clicking it a window opens to edit the steps' properties. Alternatively a right click on the icon and on "Edit step" in the context menu can be done.

Two steps can be connected via hops. The later step can then access data coming from previous step. A connection can be established by clicking on the first step and while holding the left mouse button moving over the second step.

If a hop is already established it can be edited, disabled and deleted by right clicking the hop. It is also possible the change orientation of the connection if needed.

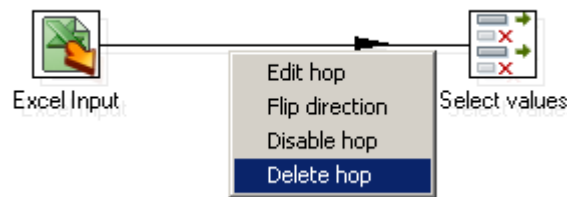


Figure 2-2 Deleting an established hop

Besides the icons to open/save the current transformation or job in the bar above the workspace the icons for running, pausing and stopping a transformation are located. There are also possibilities to create a preview and debug the transformation.



Figure 2-3 Control icons of a transformation

In a job there is also an icon to run and stop the current job.



Figure 2-4 Control icons of a Job

These are the very basic commands that are needed to create transformations and jobs with Pentaho Kettle Data Integration. Everything that is required for individual transformation steps or job entries is explained in the following sections when needed.

3 Pentaho Kettle Data Integration - Transformations

Here the most important transformation steps are explained and exemplified.

3.1 Input Steps

The category Input Steps provides steps that are used to import data from many different kinds of sources. The most important will be shown with examples.

3.1.1 Excel Input

Data from an Excel sheet can be imported using the Excel Input transformation step. When double clicking the icon the window in Figure 3-1 opens. Here one or more Excel Files can be added from the local file system. When clicking the "Add" button the file in the "File or directory" text box is added to the selected files.

It is also possible to get filenames from a previous step. Therefore the appropriate check box has to be selected and the name of the step the fields that contain the filenames have to be chosen. A possible step to get file names from is "Get File Names". Its function is explained later in section 3.1.3.

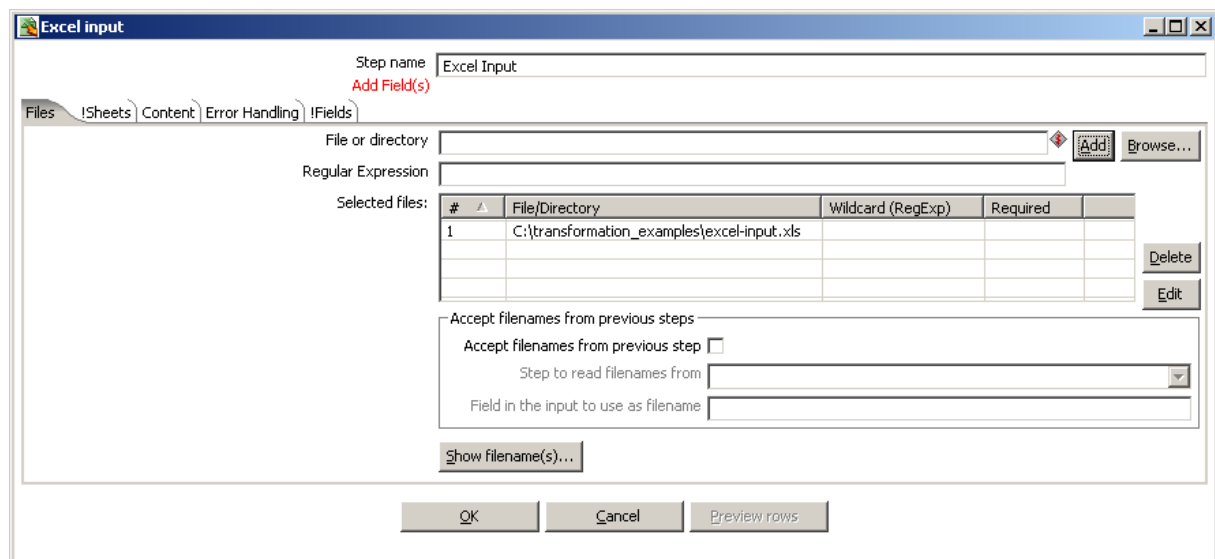


Figure 3-1 Choosing Excel file(s) to import

Then in the "!Sheets" tab the sheets that should be used can be selected. With the "Get sheetname(s)" button individual sheets of the selected Excel files can be selected. To choose all sheets just enter zeros for "Start row" and "Start column" in the first line.

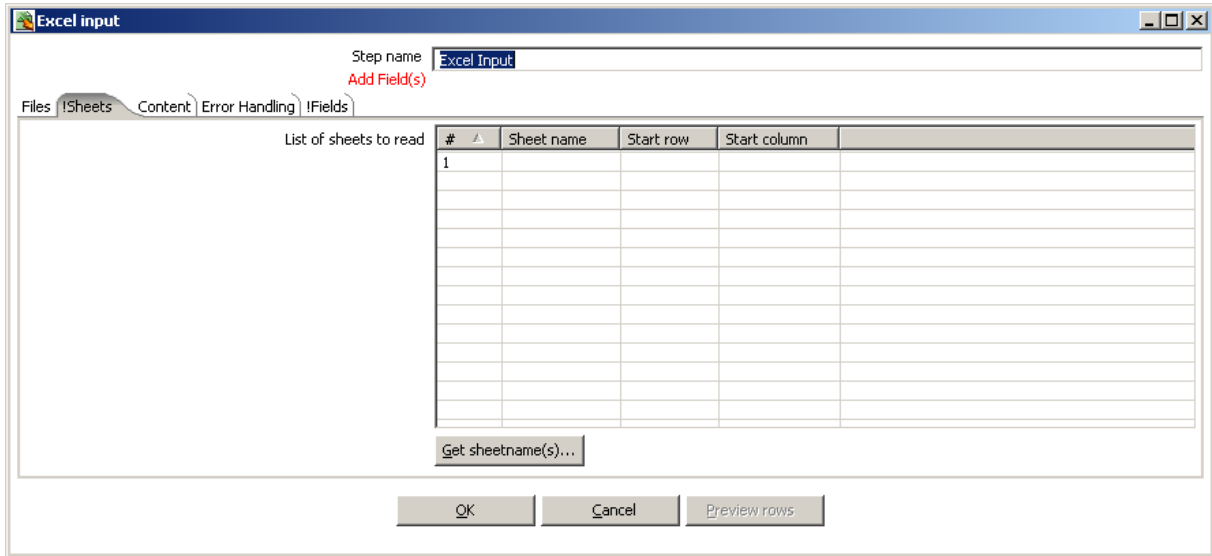


Figure 3-2 Selecting sheets from an Excel file

Finally in the tab "Fields" the fields to select can be chosen. By clicking the "Get fields from header row..." button all fields in the Excel files are imported into the list. Those not required can be deleted by selecting them and pressing "del" key. Attributes as Type, Length and Precision can be edited by clicking these fields. It is very important to add the right date format when dealing with fields with dates in it. This can be entered in the Format fields (e.g. dd.MM.yyyy).

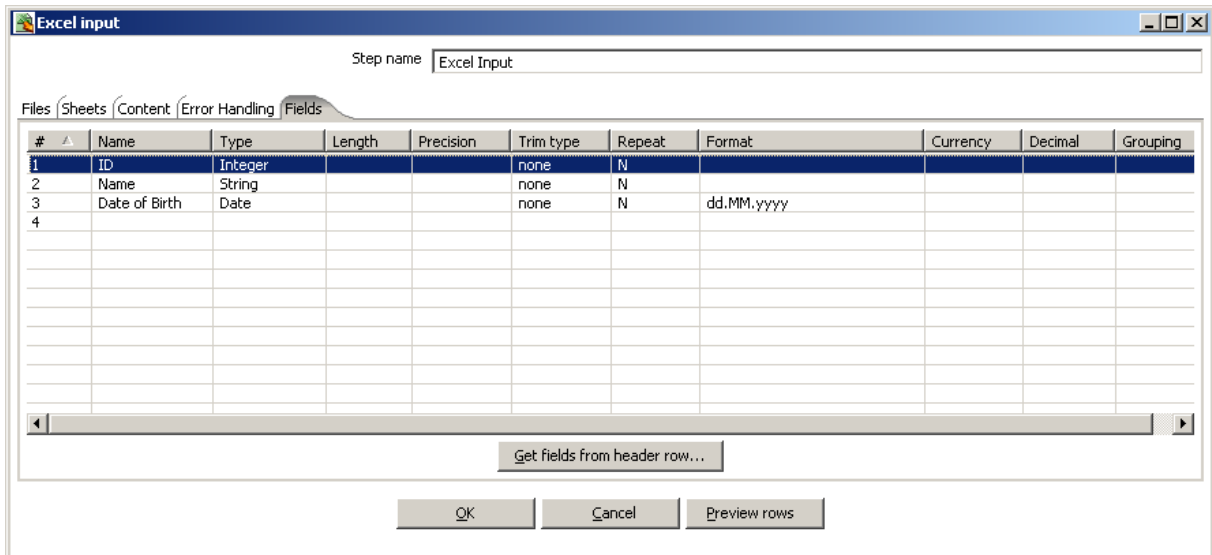


Figure 3-3 Selecting fields from an Excel File

It is possible to create a preview of the resulting rows by clicking "Preview rows". The number of preview rows can be entered and a window as in is Figure 3-4 shown.

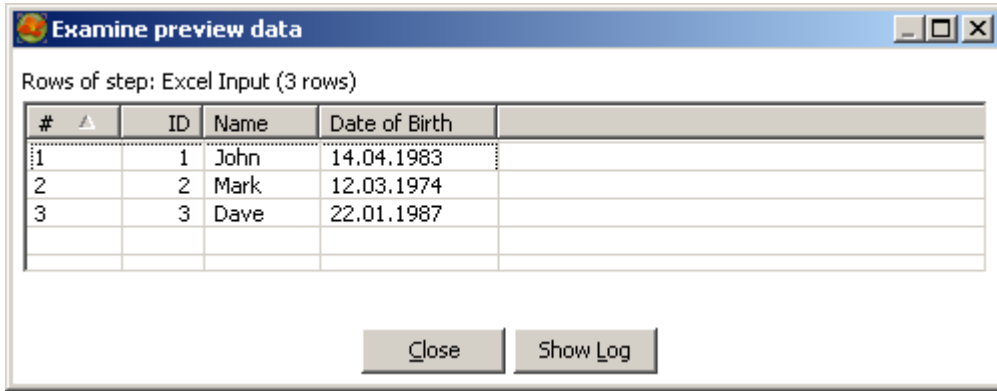


Figure 3-4 Preview of rows from Excel Input

3.1.2 Generate Rows

Like in the "Add constants" step with the "Generate Rows" step constant columns can be created. The "Limit" field determines how many rows the output will consist of.

In the list below the names and values of the fields can be entered. The example input in Figure 3-5 produces an output as seen in the preview (Figure 3-6).

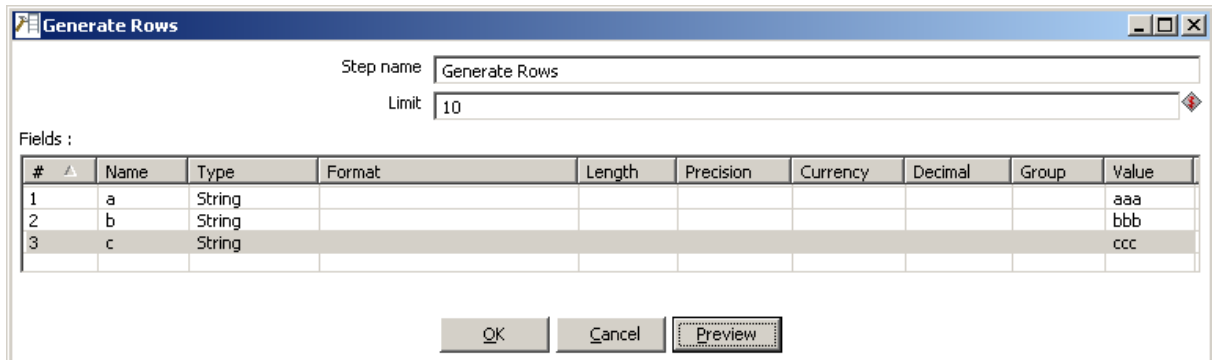


Figure 3-5 Generate Rows properties

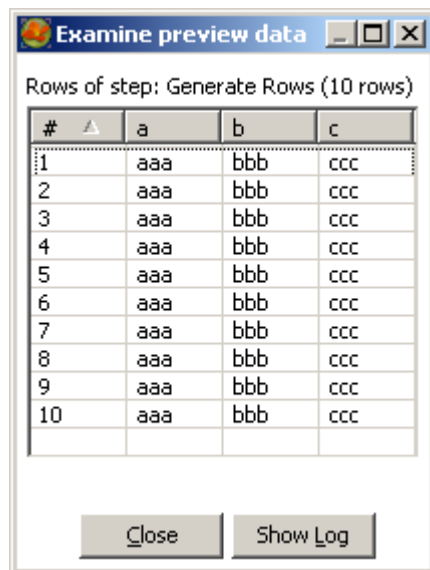


Figure 3-6 Preview of the output of the "Generate Rows"

3.1.3 Get File Names

With this step file names can be retrieved for example from a local directory. The directory can be chosen with the "Browse..." button and a regular expression can be entered in the appropriate text box. In the example from Figure 3-7 all xls files are selected from directory "C:\transformation_example" by adding the RegExp ".*\.xls". Of course this can be modified for example to get all "txt" or "xml" files. To get "xml" and "xls" files simply to entries have to be added to the "Selected files" list.

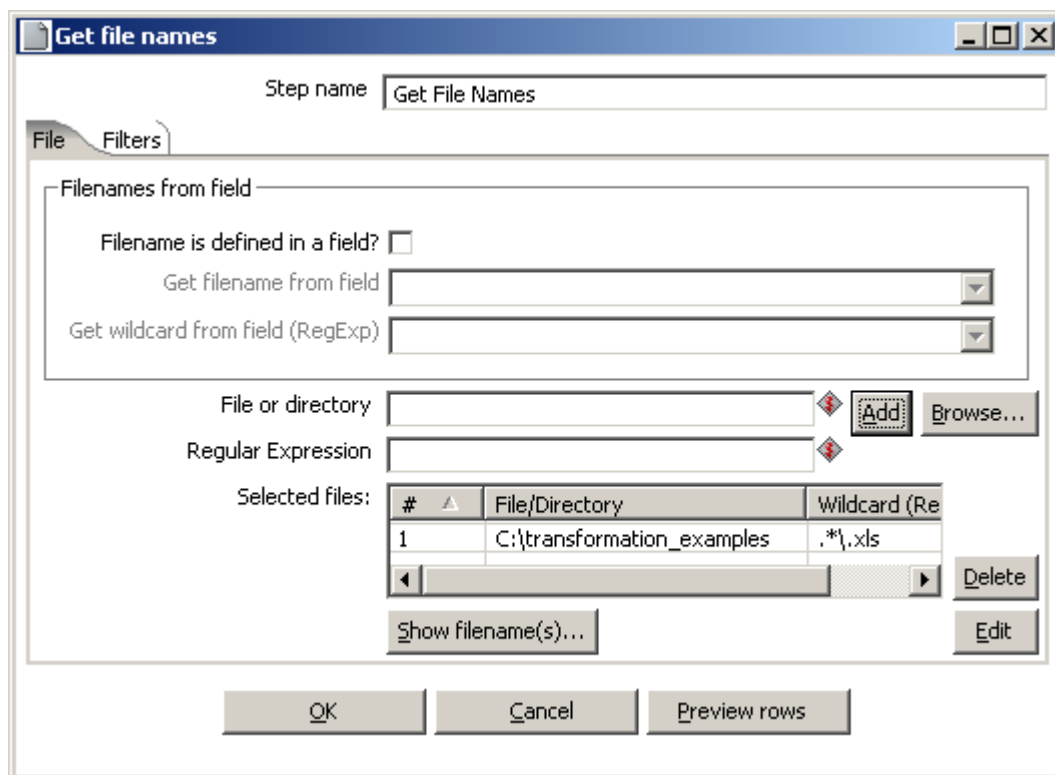


Figure 3-7 Getting file names

When previewing the rows it can be seen that the information of the path of the files are held in field filename. This field name has to be used for example when retrieving information of file names from "Get File Names" in an Excel Input as described in 3.1.1.

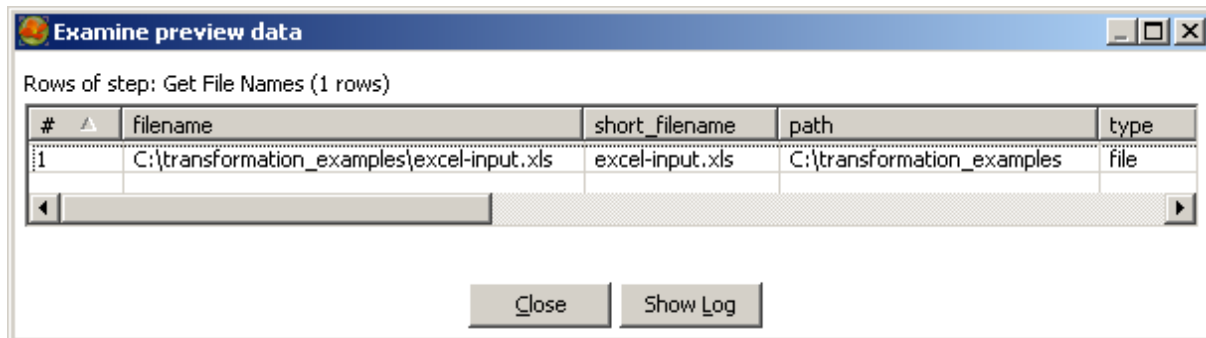


Figure 3-8 Field filename with path of files

There are a few options that can be set in the tab "Filters". Get can be "All files", "Only files" or "Only folders" depending on what is desired to get from the selected paths. Moreover a limit of files to get can be set. Default value is 0 which means that there is no limit.

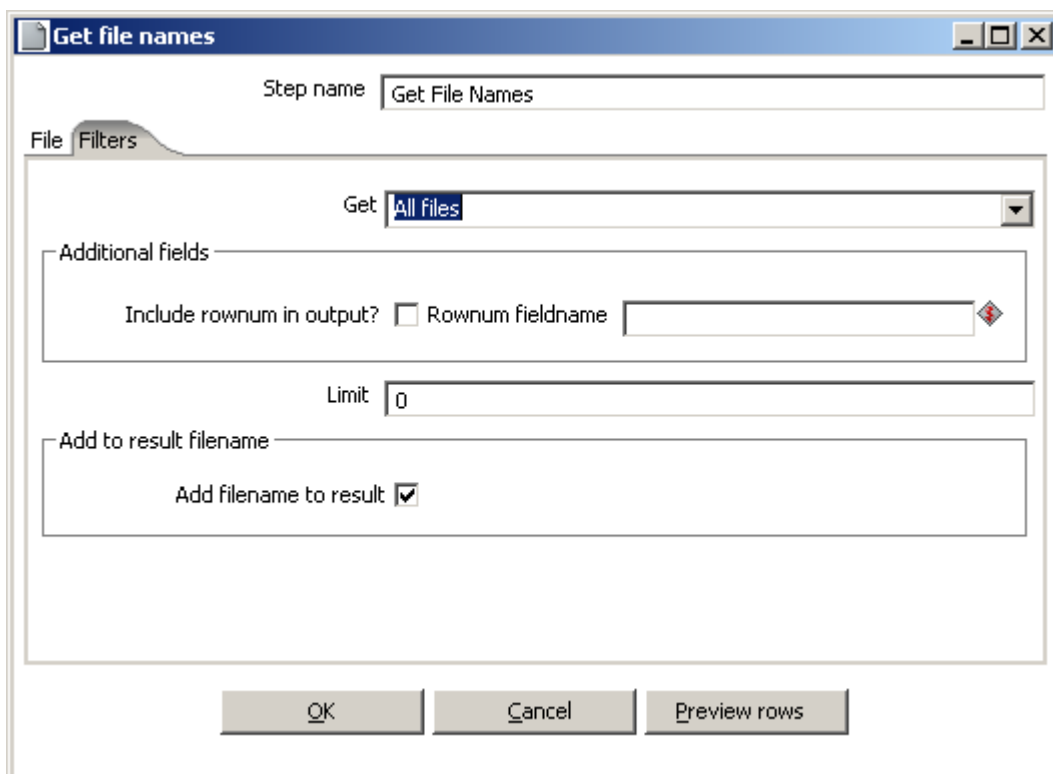


Figure 3-9 Filters of step "Get File Name"

3.1.4 Get data from XML

In this step information can be imported from an XML file. XPath statements are used to retrieve the right data. As an example a very simple XML file is used:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <example_xml>
- <element>
  <element_name>first</element_name>
</element>
- <element>
  <element_name>second</element_name>
</element>
- <element>
  <element_name>third</element_name>
</element>
- <element>
  <element_name>forth</element_name>
</element>
</example_xml>
```

Figure 3-10 Example XML file for "Get data from XML" step

The files tab is similar to the one of the Excel Input step (3.1.1). The path to the file can be chosen directly or from a field coming from a previous step.

In the content tab the "Loop XPath" field is very important. Here the repeating element can be entered. In this case the path to the repeating element is "/example_xml/element". That is because the values in the "element_name" tags are desired. A path like "/example_xml/element/element_name" would not produce any fields, because there are no more elements below the "element_name" tag. There is a very simple way to see which XPath statements can be used. Click the "Get XPath nodes" button and a window as in Figure 3-12 will appear. Here a possible XPath can be chosen.

After entering this information in the "Fields" tab the desired fields can be selected. A click on the "Get fields" button lists the possibilities. Those not desired can be deleted from the list. In order to retrieve the values of the "element_name" elements it is listed there.

The preview should be as in Figure 3-14.

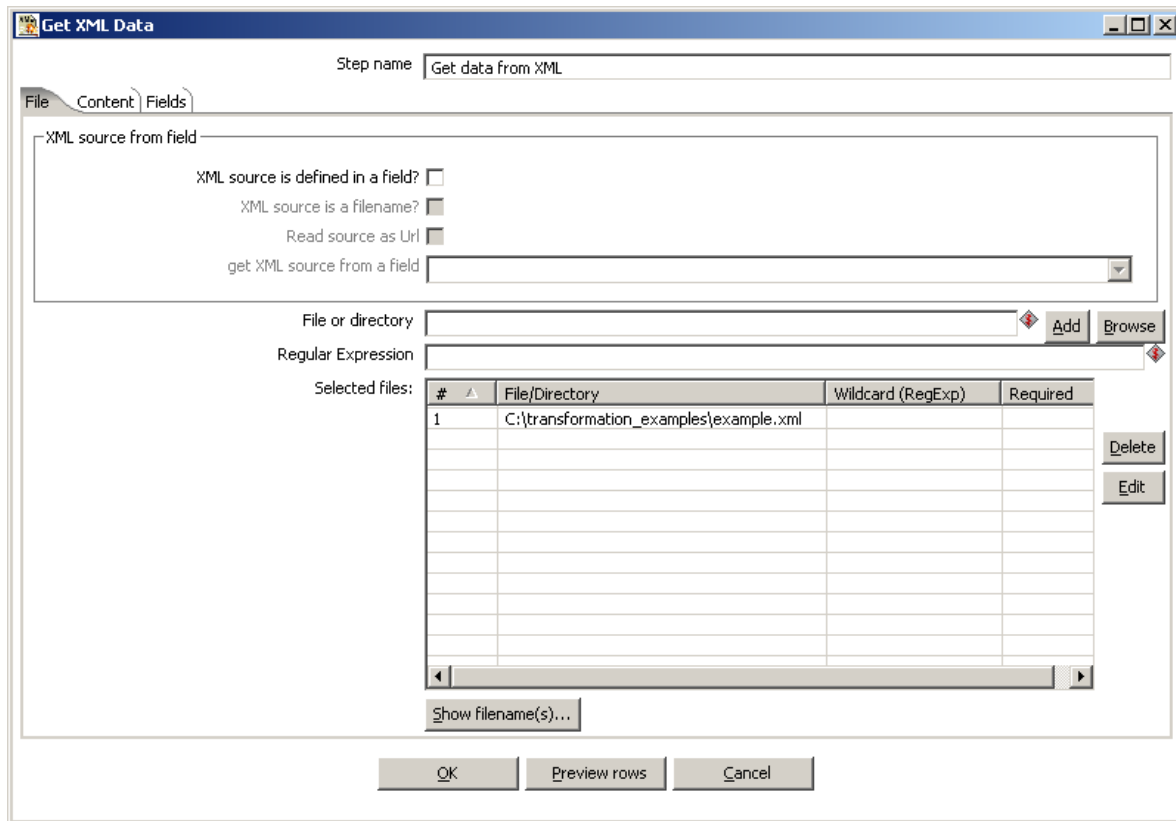


Figure 3-11 Get data from XML properties (File)

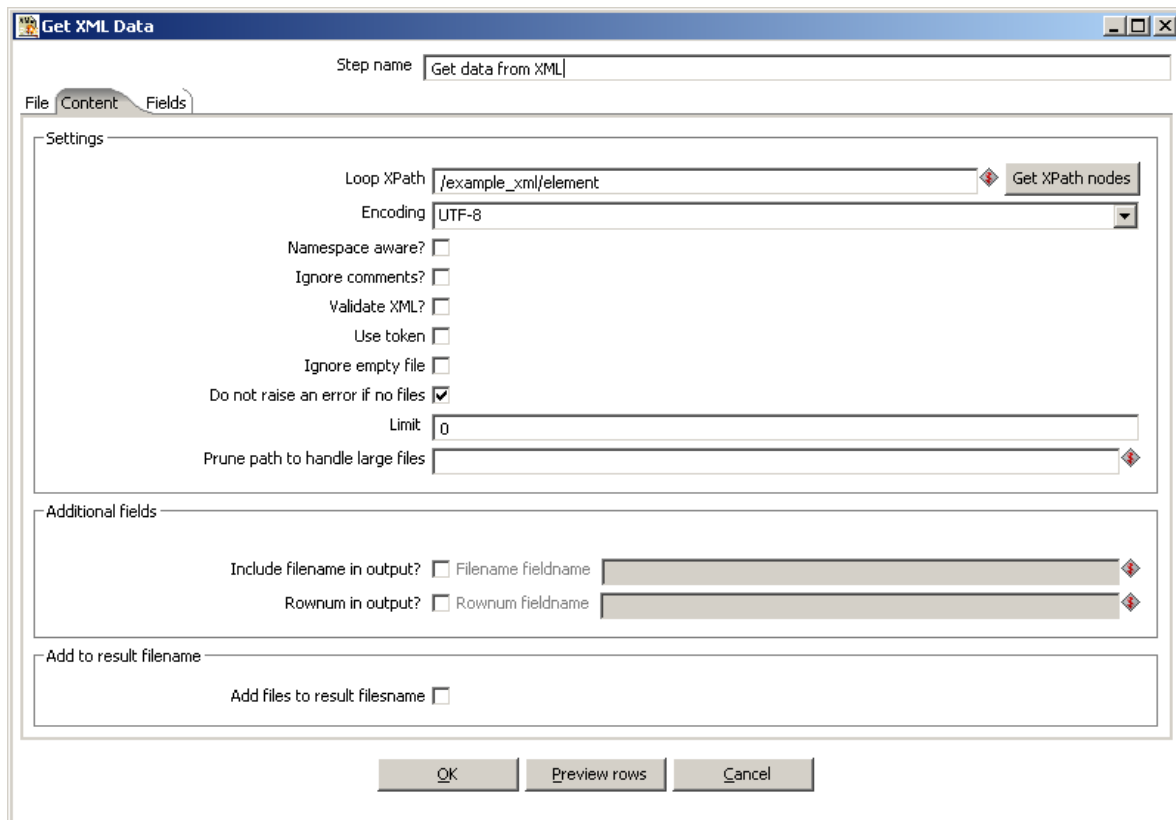


Figure 3- Get data from XML properties (Content)

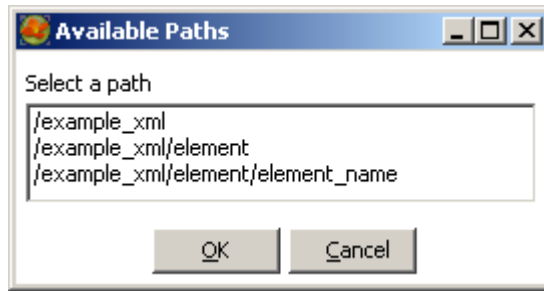


Figure 3-12 Get data from XML (Available Paths)

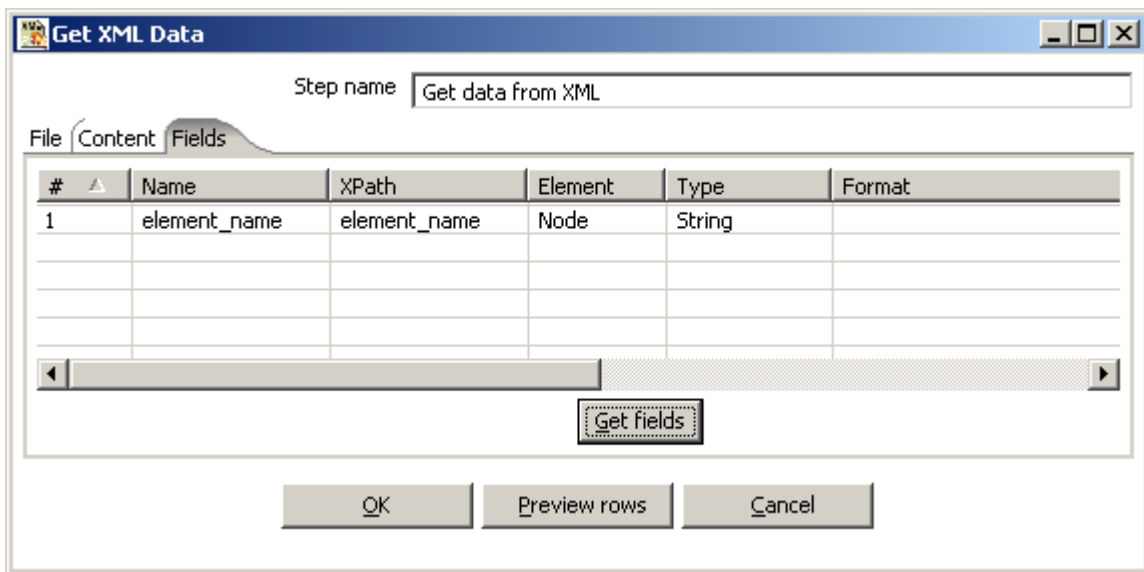


Figure 3-13 - Get data from XML properties (Fields)

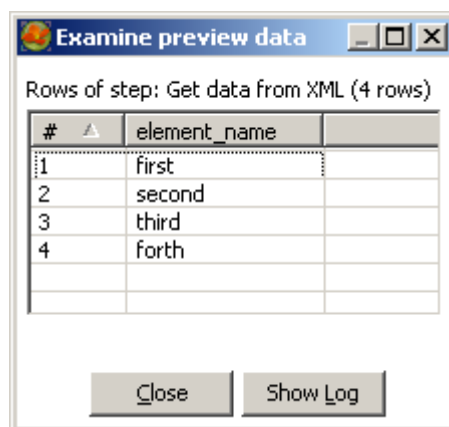


Figure 3-14 Preview of "Get date from XML" step example

3.1.5 Table Input

This step gets information from database tables. In the properties of this step a new connection to a database can be created by clicking the button “New...” as seen in Figure 3-15. The window in Figure 3-16 appears.

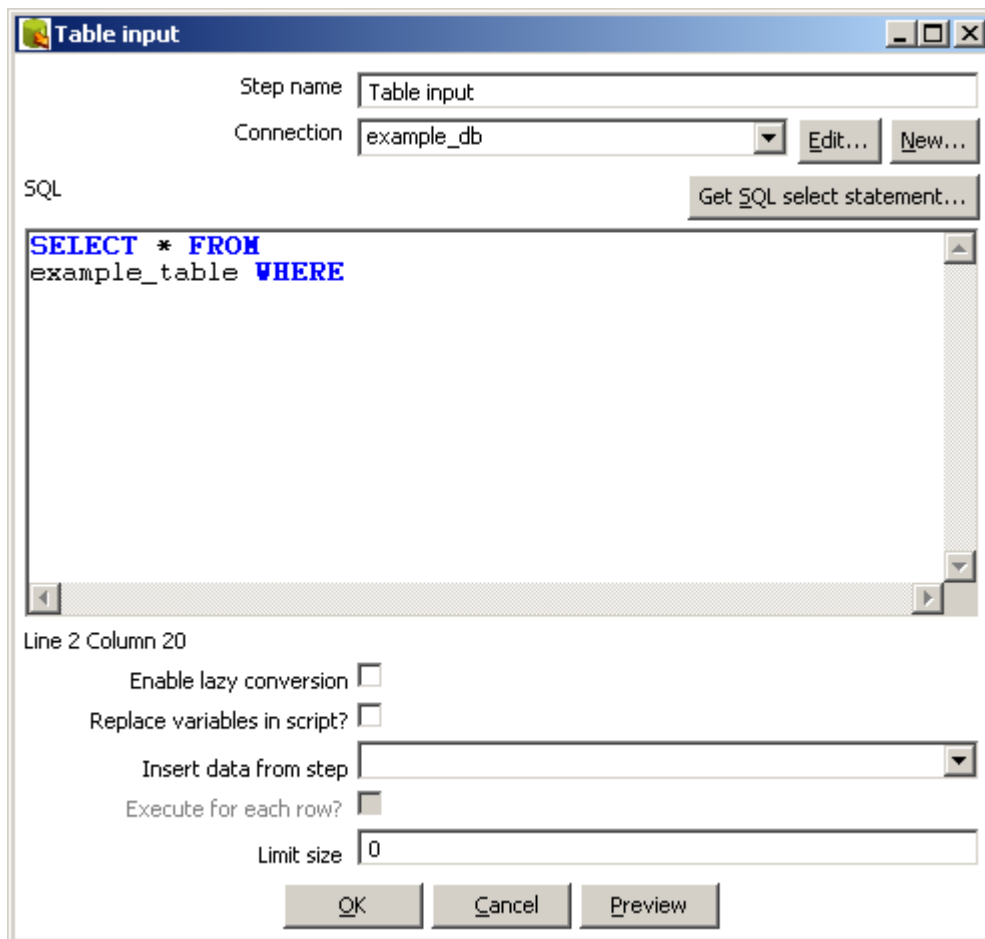


Figure 3-15 Table input properties

The example shows a connection to a MySQL database. The connection type can be selected in the scroll box. In the “Settings” field the connection settings can be entered. The database name can be extended with additional parameters like `example_db?zeroDateTimeBehavior=convertToNull&useUnicode=true`. This statement for instance is very useful if timestamps with all zero values appear in a table. These are converted to Null values in order to be able to process them.

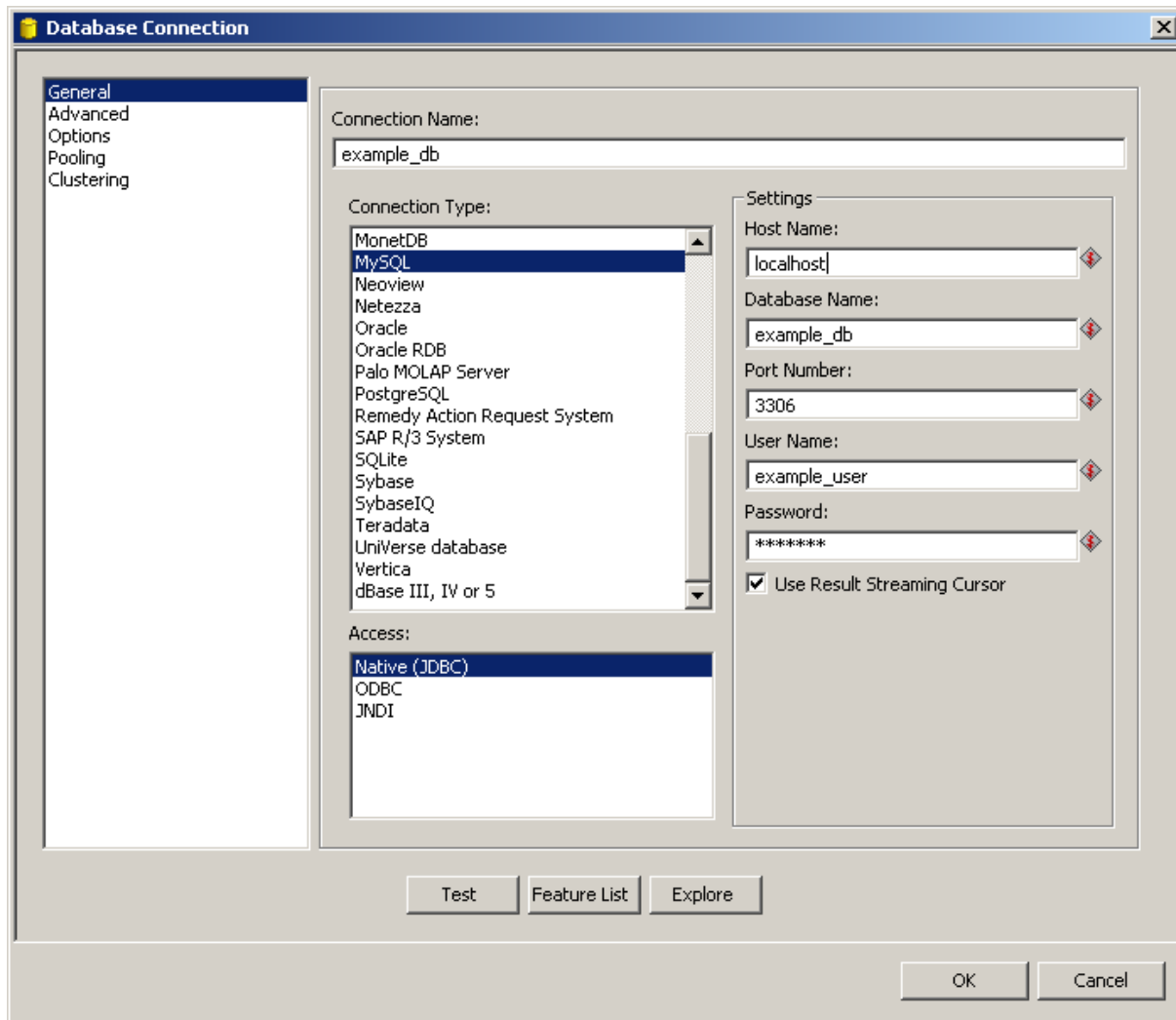


Figure 3-16 Establishing a database connection

3.2 Output Steps

This category contains steps that can be used to create output data in various formats. An output step receives data from a previous step and can then access it. Figure 3-17 shows a very simple transformation consisting only for getting input values selecting certain values and then creating an XML output. The "Select values" step will be explained later in section 3.3.6.



Figure 3-17 Simple transformation

Again the most important will be illustrated in the this section.

3.2.1 Excel Output

With the Excel output step transformed data can be stored in an Excel file. Figure 3-18 shows the file options of this step. A file name and path has to be specified. The extension is xls by default. There are several other options that can be activated by checking the check boxes.

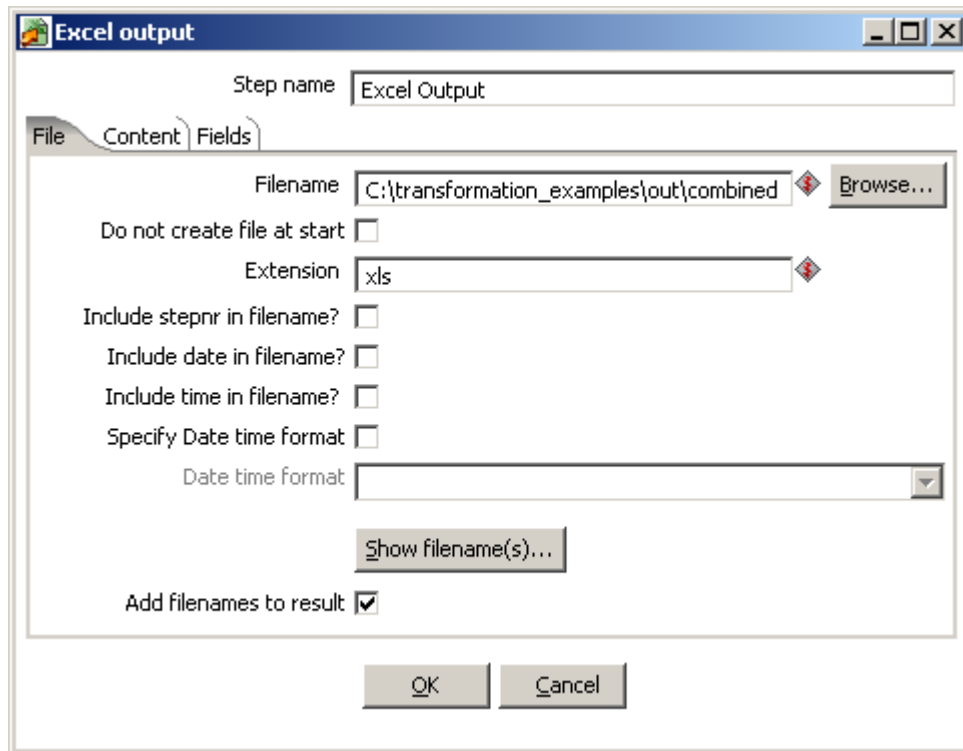


Figure 3-18 Excel output properties

In the "Content" tab the encoding can be set. It is also possible to split the output into several files when entering a number in the "Split every ... rows" field. The sheet name can be chosen and selected if the sheets should be protected or not. If the output sheet is likely to be viewed by someone, it is handy to select the "Auto size columns" option. This avoids too small columns in the output Excel file, which would be represented by "#####" to the user.

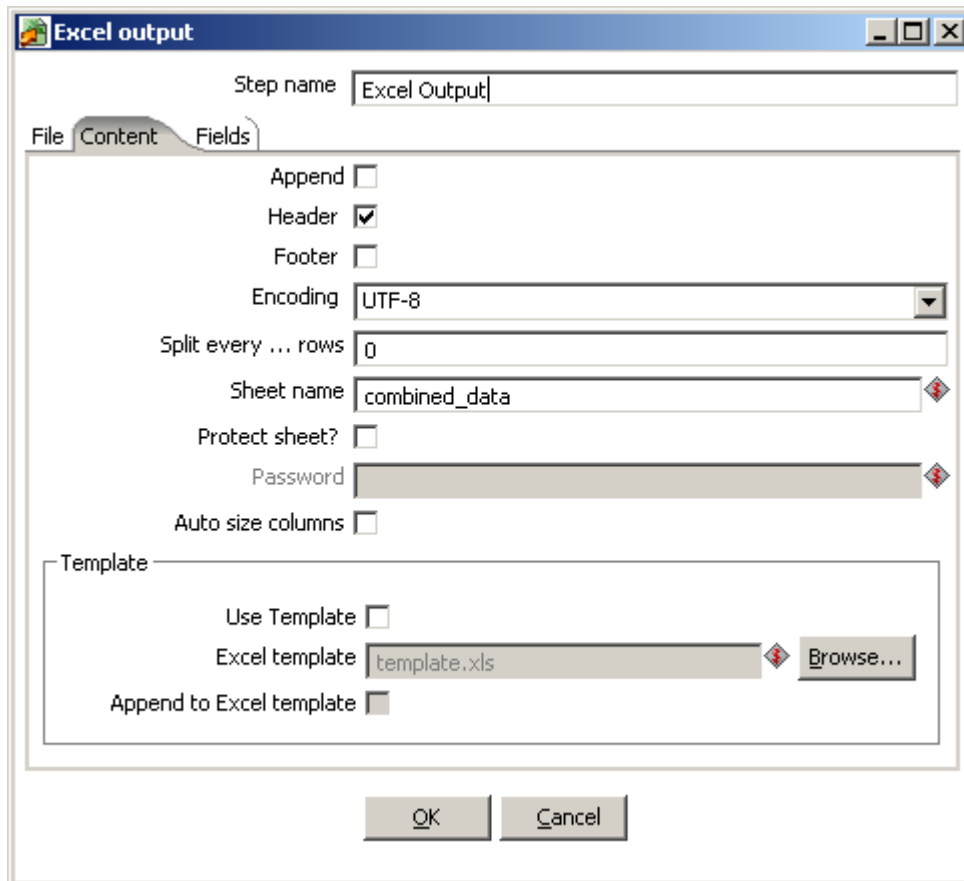


Figure 3-19 Editing content of the Excel output

Finally the fields to be written into the Excel sheet have to be selected. The button “Get Fields” shows all possible fields. They can be removed, or changed in type and format.

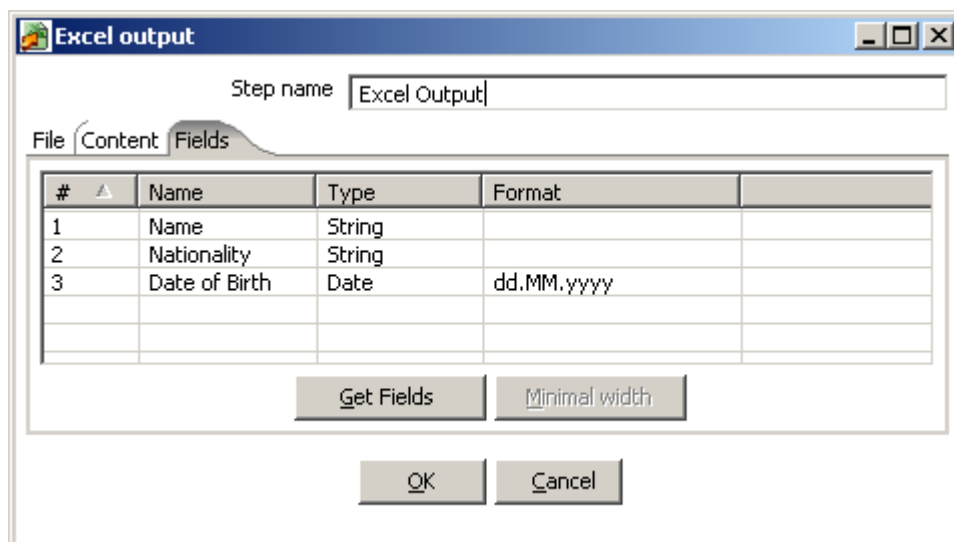


Figure 3-20 Selecting fields for Excel Output

An example of using this output step is illustrated in section 4.7.

3.2.2 Text file output

This step produces a text file from one or more fields. For instance it can be used to write a xml field into a file. In the "File" tab the path of the output file and its extension can be specified. In this case xml is chosen because a XML file will be created from a XML field coming from a previous "XML add" as in section 3.3.1.

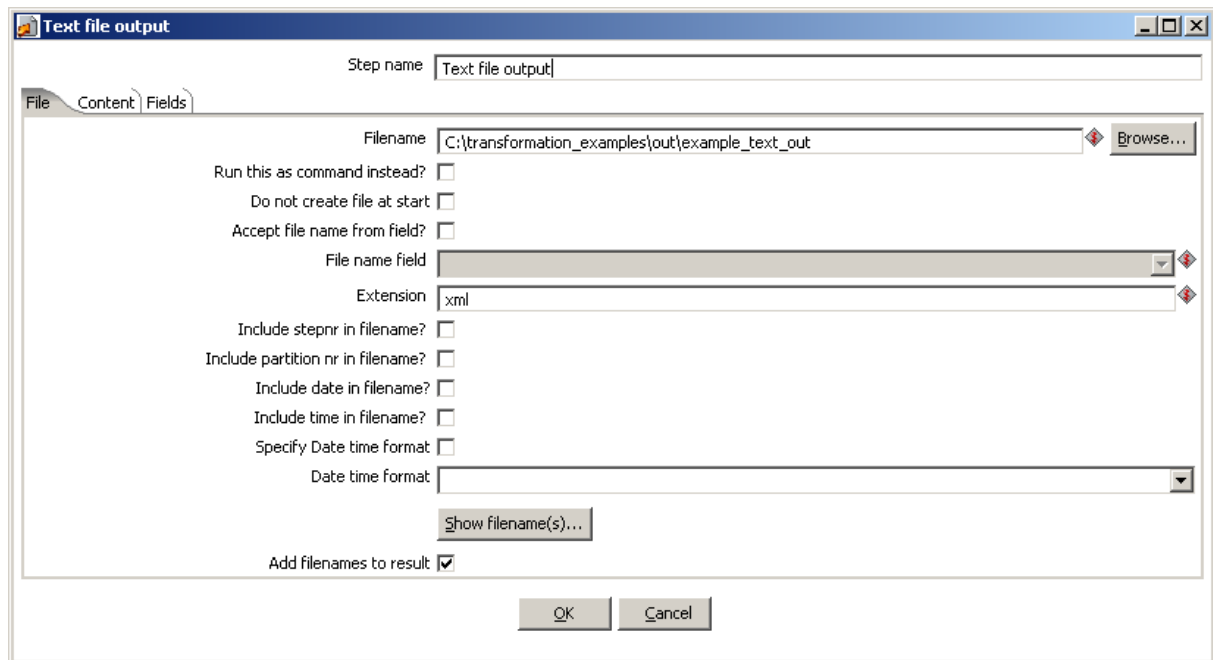


Figure 3-21 Text file output properties (File)

In the "Content" tab some options can be set. If the "Append" Check Box is selected, the original content of the file (if it exists) is not deleted, but the new content is appended to that already in the file.

Another important option is "Split every ... rows". If set to 1, an individual file is created for every row of the input.

In the last tab "Fields" the fields that are listed are included in the output text file. If more fields are selected in the "Content" tab a separator can be set. For example if ";" is used as a separator the elements of two fields are separated like this: element1;element2.

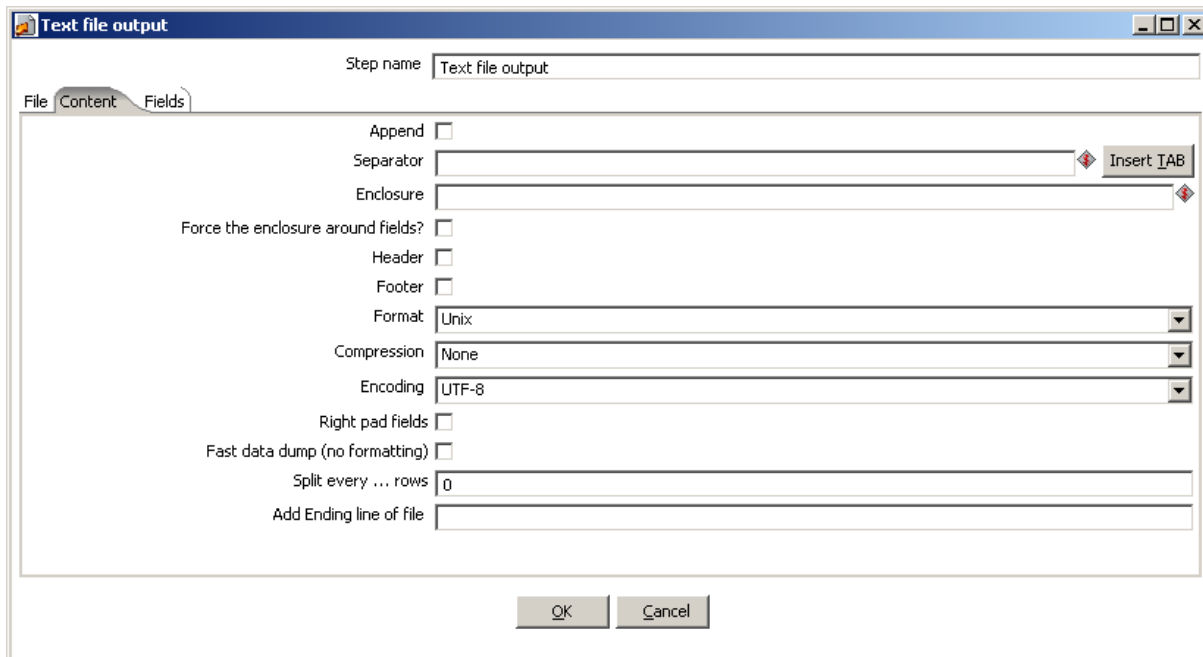
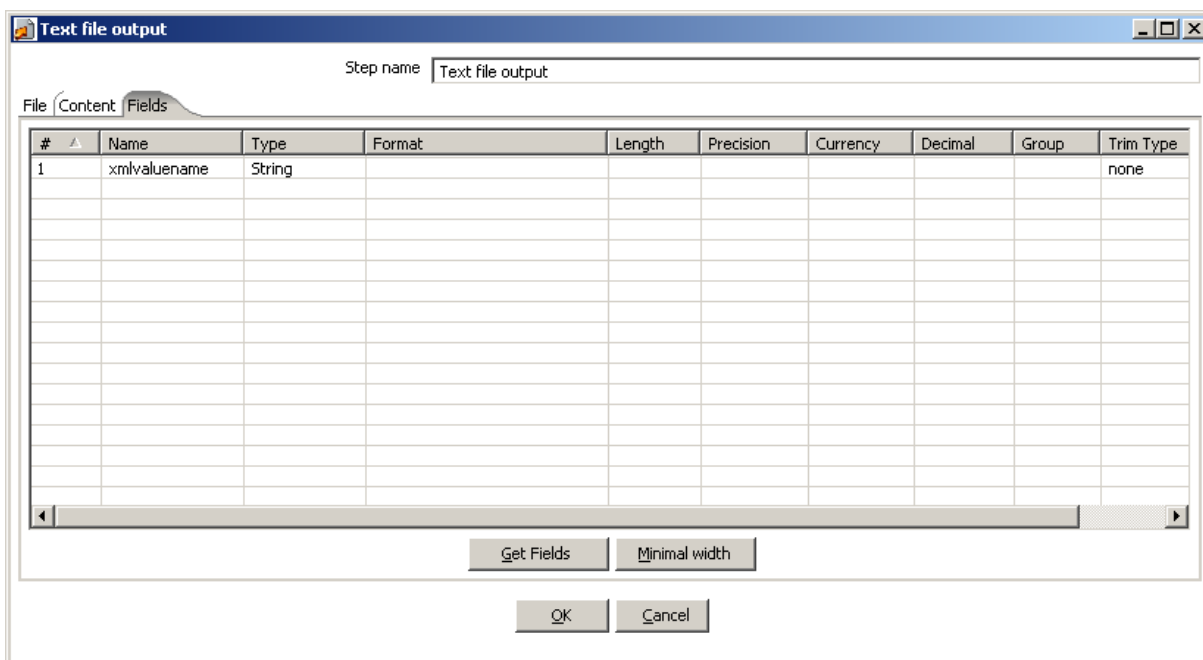


Figure 3-22 Text file output properties (Content)



3.2.3 XML Output

The XML output step is used to create one or more XML files from given input (coming from previous steps). The path and filename of the output file(s) can be entered in the "Filename" text box or chosen with the "Browse..." button. The extension is xml by default, but can be changed in the appropriate text box. Options like including the step number, date or time in the file number can be checked optionally. The date and time format can be chosen or entered in the combo box if the checkbox is checked.

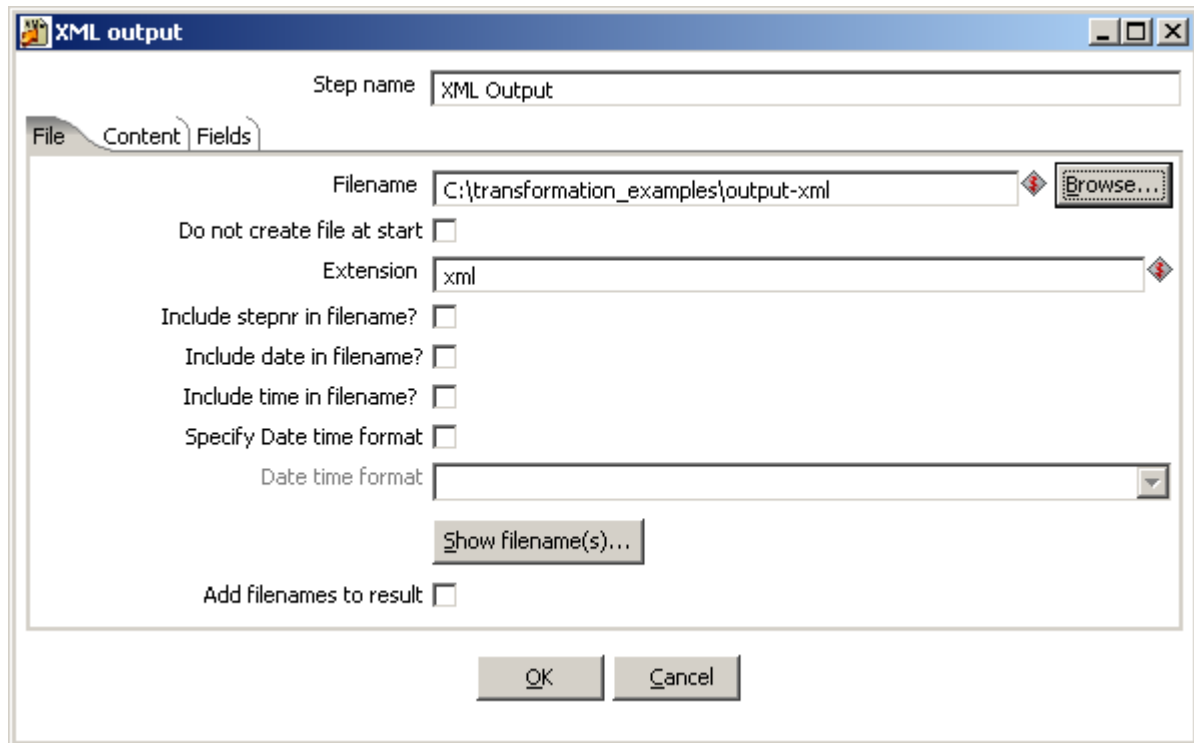


Figure 3-23 XML output properties

In the "Content" the encoding of the output file(s) can be set. The parent XML element's and the row element's names can be set too.

The "Split every ... rows option creates single xml files every row if 1 is entered, every two rows if 2 is entered and so on. If 0 is entered only one xml file will be produced as output.

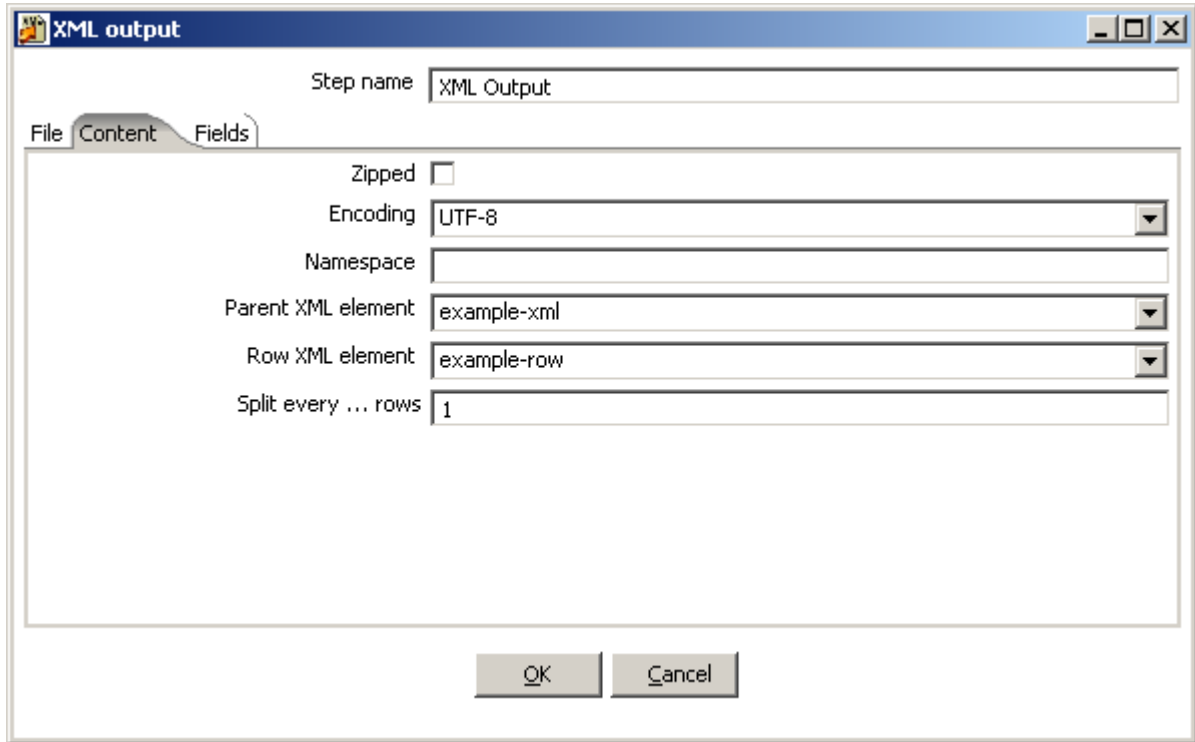


Figure 3-24 Editing the content of the XML output

In the last tab of this step "Fields" the fields for the output can be selected. The "Get Fields" button shows all possible fields in the list above. Undesired fields can be removed by simply selecting them and pressing the del key. For every field name a different element name can be entered. If the "Element name" is left blank for a field, its field name is used as element name as well.

Again here "Type" and "Format" can be edited. For instance a date format as dd.MM.yy can be entered.

Additional options are to set values for "Length" and "Precision" of the field values.

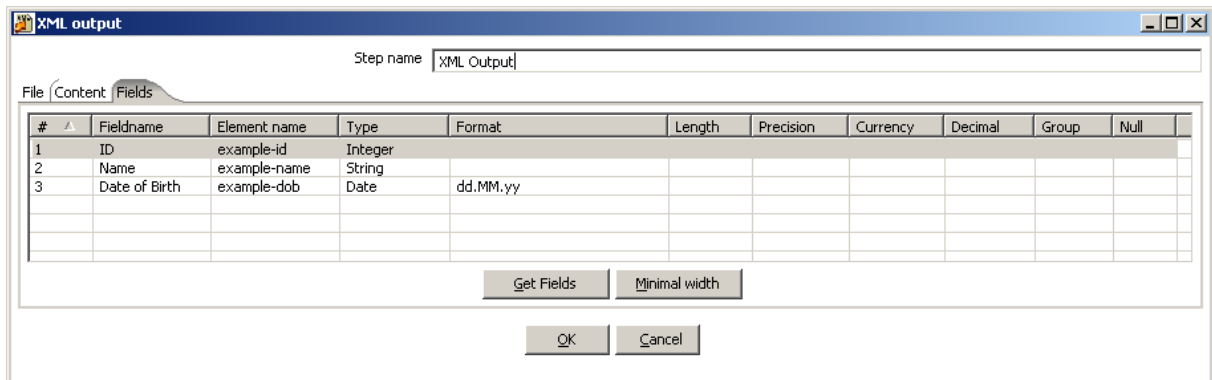


Figure 3-25 Selecting fields of XML output

As an example a transformation as in Figure 3-17 is run. The "Select values" step has no effect in this example.

In the "Excel input" step all fields of the Excel file are selected. The input is just a simple set of three fields ID, Name and Date of Birth with three values each (Figure 3-26).

The output step is configured as in the figures above in this section. Because "Split every ... rows" is set to 1 the transformation produces three individual xml files as shown in Figure 3-27.

	A	B	C
1	ID	Name	Date of Birth
2	1	John	14.04.83
3	2	Mark	12.03.74
4	3	Dave	22.01.87

Figure 3-26 Excel input

```
<?xml version="1.0" encoding="UTF-8" ?>
- <example-xml>
- <example-row>
  <example-id>1</example-id>
  <example-name>John</example-name>
  <example-dob>14.04.83</example-dob>
</example-row>
</example-xml>
```

Figure 3-27 Sample Output of the XML output step

3.3 Transformation Steps

Transformation steps offer various possibilities to change data from previous steps or for example add constant values to the transformation.

The following steps in this section are connected to the Excel Input from above (Section 3.1.1) if needed and not explicitly mentioned differently.

3.3.1 Add XML

An XML fragment can be created from a number of fields from the input stream using the Add XML step.

In the "Content" tab of the step's properties the name of the output field can be set. This field will contain the created XML data as a string. This string can for example be saved as a XML file using the "Text file output" step (3.2.2) by setting the extension "xml" and the outgoing field of this step as an output value of the "Text file output" step.

The name of the root element can be specified and checked if the XML header should be omitted or not. If this option is checked `<?xml version="1.0" encoding="UTF-8" standalone="no" ?>` would not be in the output for instance.

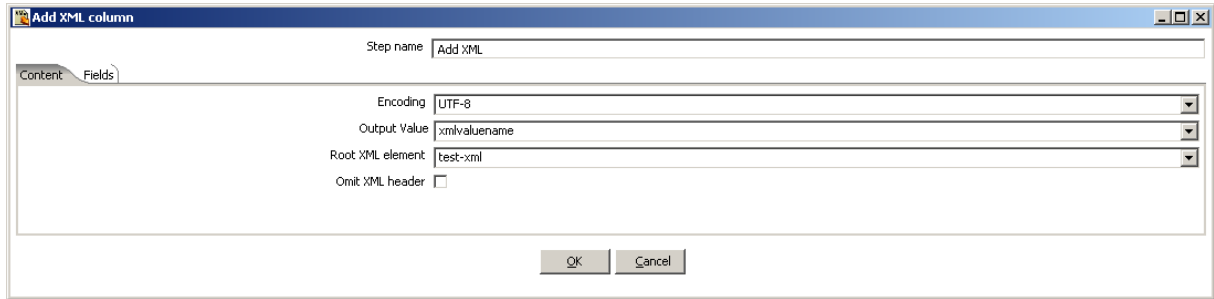


Figure 3-28 Editing the content of the "Add XML" step

In the fields tab the XML elements and attributes can be set. In the "Fieldname" field the names of the incoming fields must be entered. If desired an element name can be chosen for the fields. If left blank the field name is taken by default.

If "N" is entered in the field "Attribute" a field represents a XML element. If "Y" is chosen the field is an attribute and its parent element can be entered in the next column. If "Attribute parent name" is left blank, the field will become an attribute of the root element.

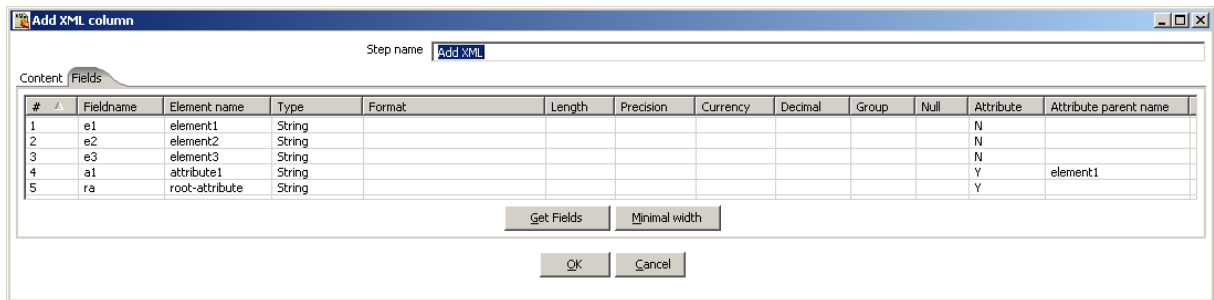


Figure 3-29 Choosing fields for XML

If values are entered as in Figure 3-28 and Figure 3-29 the following XML file is created, if the resulting field from the "Add XML" step is saved in a file using the "Text file output" step.

```

<?xml version="1.0" encoding="UTF-8"
standalone="no" ?>
- <test-xml root-attribute="ra">
  <element1 attribute1="a1">e1</element1>
  <element2>e2</element2>
  <element3>e3</element3>
</test-xml>

```

Figure 3-30 Example output of the "Add XML" step (using "Text file output")

3.3.2 Add constants

Creating constant values works very intuitively. It handles no input. Simply the name of the name, type and the value of the constant have to be entered. Additional options can be set, for example a date format if a constant date is desired.

The constants can then be used in a following step (e.g. XML Output).

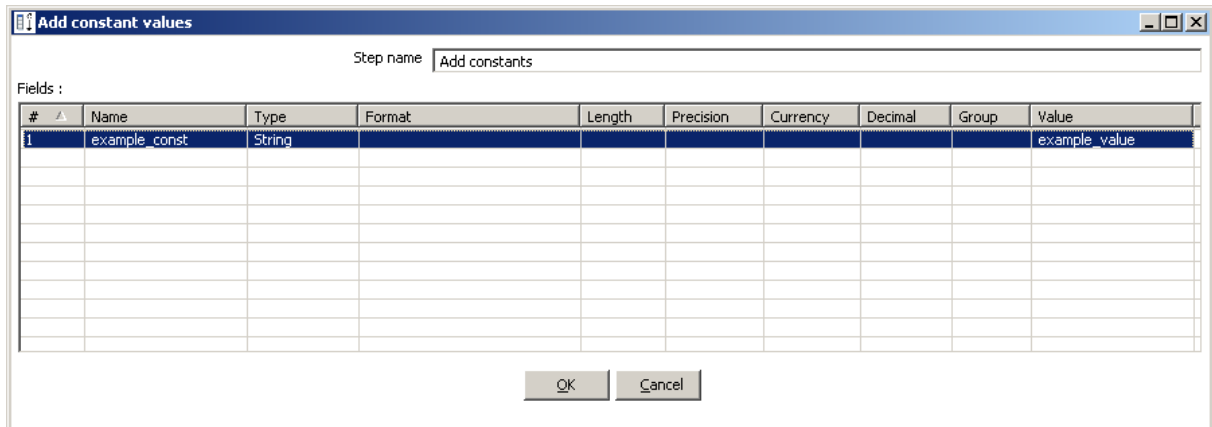


Figure 3-31 Creating constants in a Transformation

3.3.3 Calculator

The calculator has many functions listed in Figure 2-1. The name of the resulting field and its return type has to be specified. Depending on the operation that is selected in field "Calculation" in Field A,B and C the input fields must be entered.

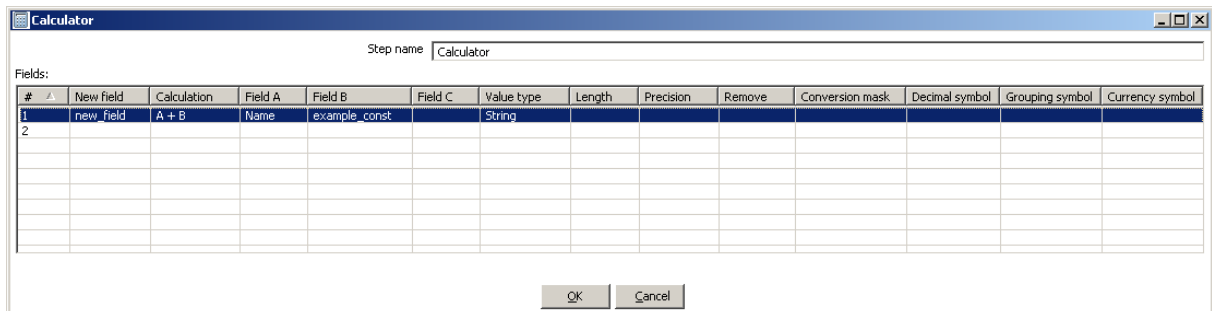


Figure 3-32 Using Calculator to merge strings

Two strings can be merged by selecting calculation A+B and return type string. Moreover there are operations for manipulation of dates like getting the year, month or week of a date and adding or removing time to a date.

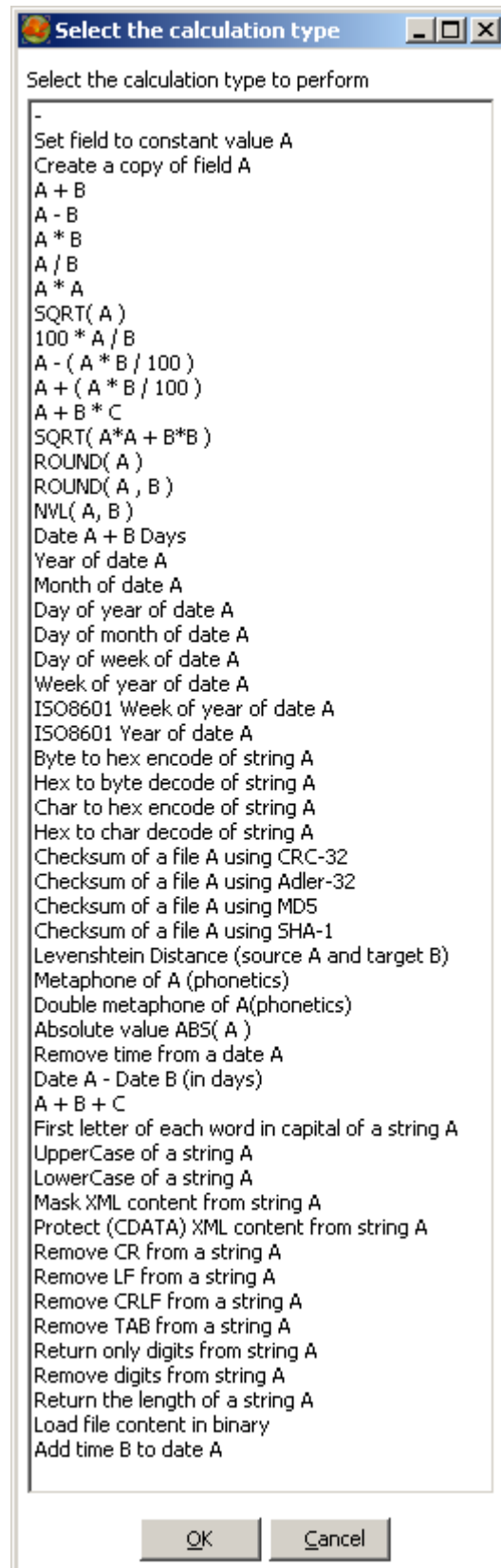


Figure 3-33 Predefined Calculator Operations

3.3.4 Replace in string

It is possible to replace parts of a string with this step. The string comes from a previous step in a field. This has to be entered in the list. If an out stream field is entered a new field is created for the new string. If this field is left blank the original string is updated.

In the "Search" field the string to replace can be entered, or if chosen a regular expression. The "Replace with" field contains the new content to replace the old one.

It can be chosen if the expressions that need to match are case sensitive or not.

The example below replaces every "_" in "incoming_string" with a "-". The result is directly stored back in the field "incoming_string".

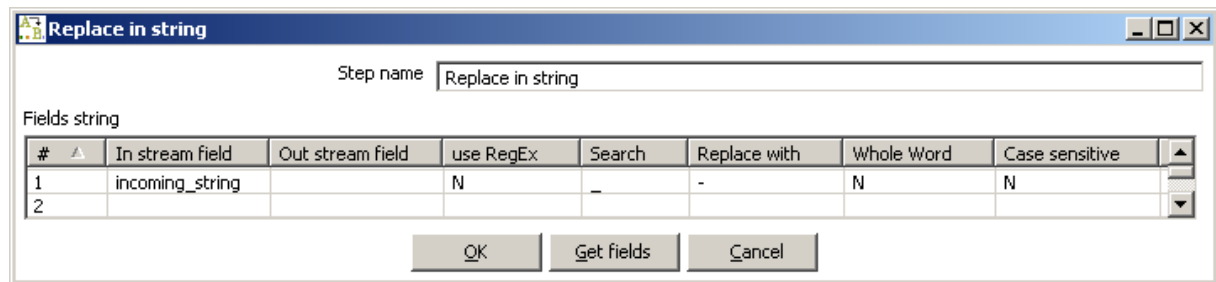


Figure 3-34 "Replace in string" properties

3.3.5 Row flattener

3.3.6 Select Values

The "Select values" step gets data from a previous step. By pressing the "Get fields to select" button all possible fields are shown. The names of the output fields of this step can be changed if needed by entering the new name in the "Rename to" field.

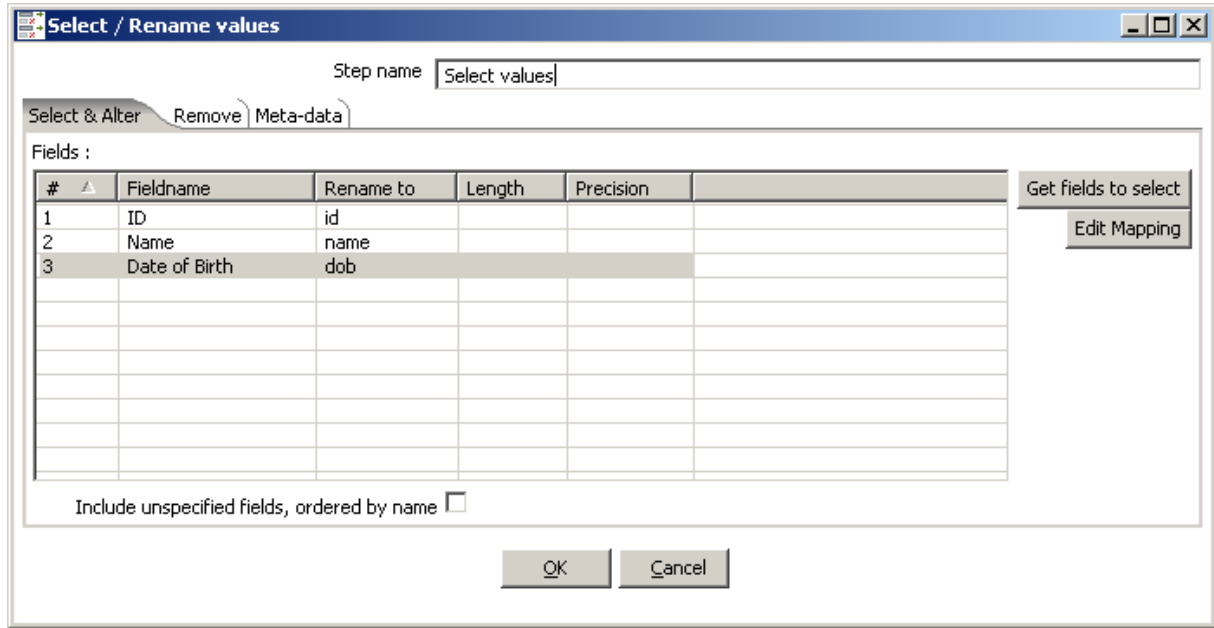


Figure 3-35 Selecting and renaming values from an input

In the remove tab unwanted fields can be deselected. For instance if the ID of the Excel input is not needed it can be listed here to remove it. The "Get fields to remove" lists all possible fields. Those that are not needed can be left in the list and are thus not sent to the next step.

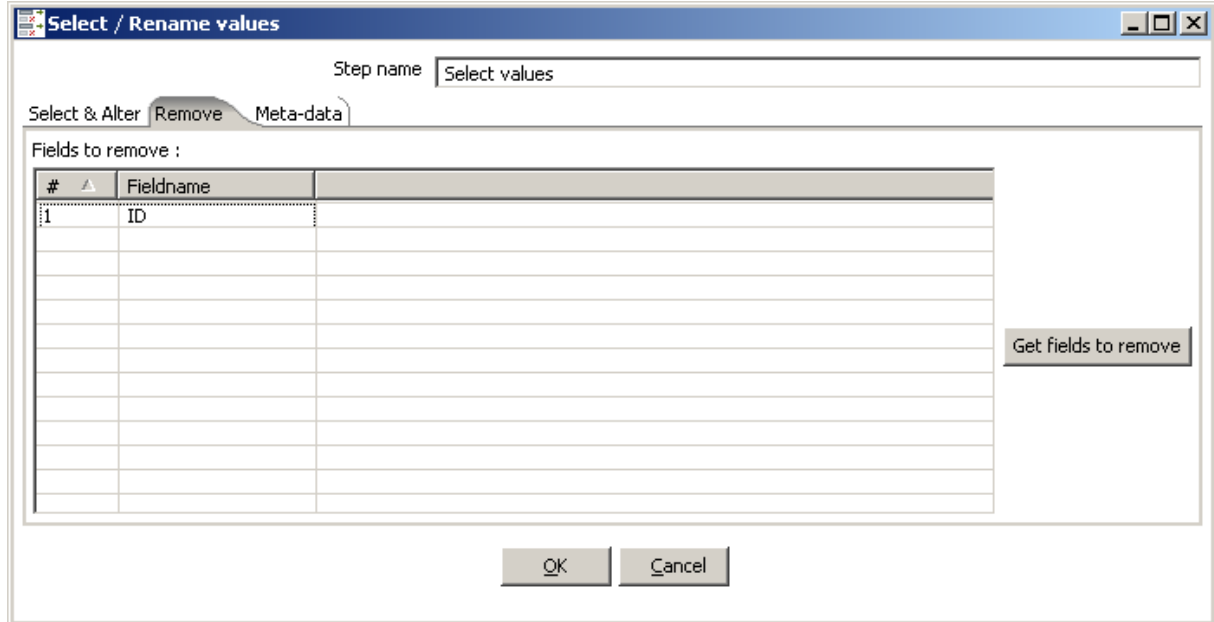


Figure 3-36 Removing unnecessary Fields

In the "Meta-data" tab similar changes as in the "Select &Alter" tab can be made to the input fields.

3.3.7 Sort Rows

Rows can be sorted by a specific field in ascending or descending order. More fields can be entered. The rows will then be sorted by this field, if the values in the first field are equal.

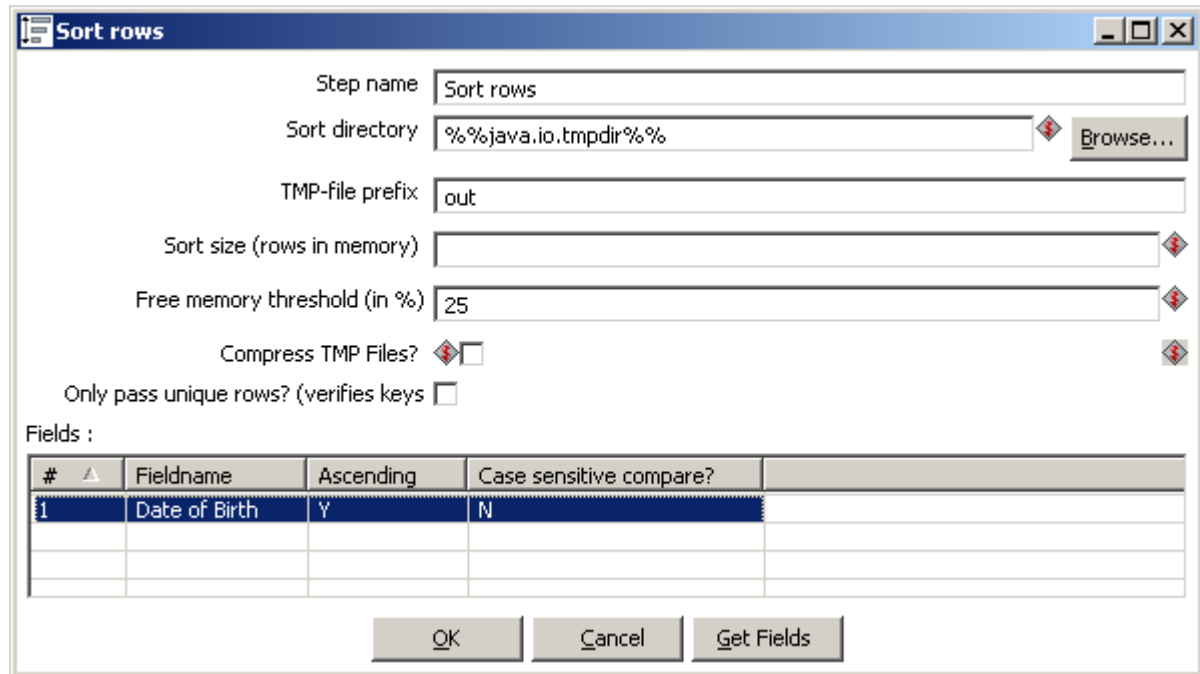


Figure 3-37 Sorting rows by a specific field

Figure 3-38 shows a preview of the Excel input sorted by Date of Birth.

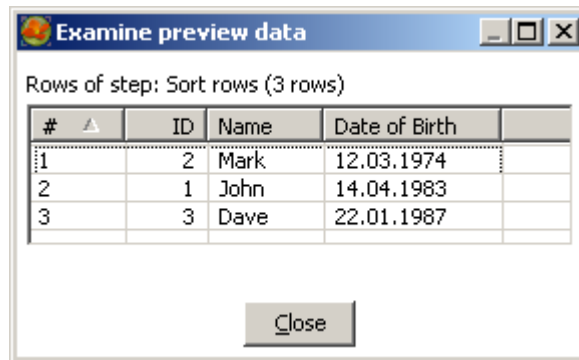


Figure 3-38 Preview of input sorted by Date of Birth

3.3.8 Split fields

This step can separate values in fields and separate them into a certain number of new fields. As an example the field Date of Birth is divided into three fields Day, Month and Year. Therefore it first has to be converted into a string. This is done in the "Select values" step as show in Figure 3-39. Because the transformation is done in the "Meta-data" tab the field Date of Birth of type Date does not exist anymore after the "Select values" step. Instead the same information is in field dob as string.

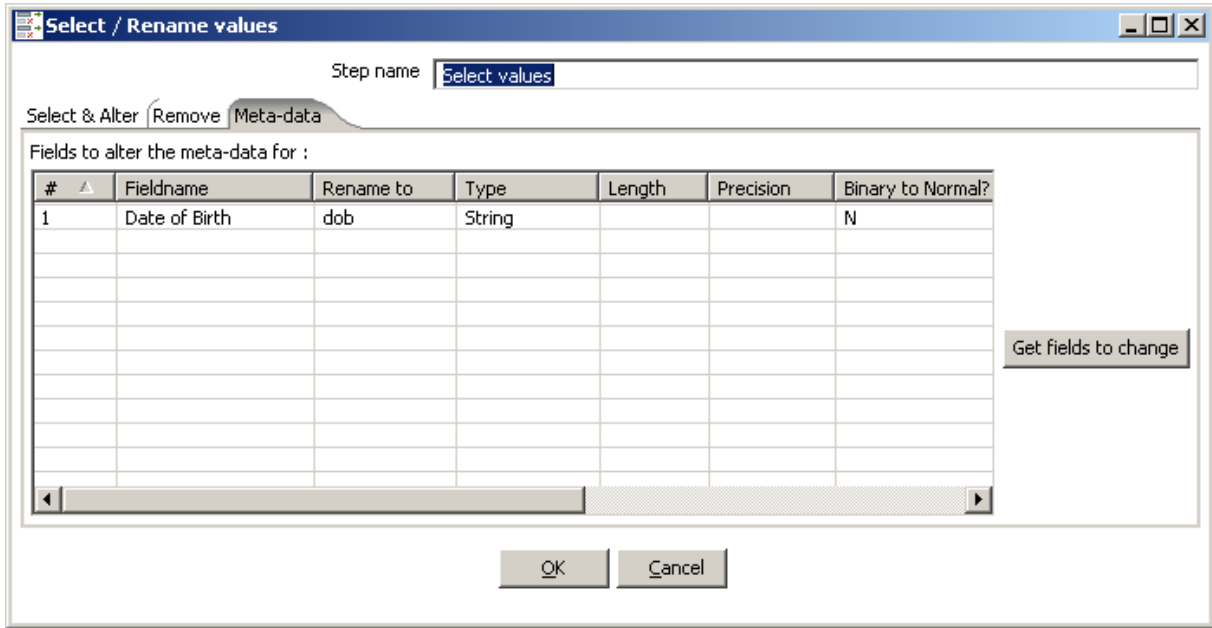


Figure 3-39 Converting Date to String (meta-data)

Then in the "File Splitter" step field dob can be selected. Because the date information is stored dd.MM.yyyy the delimiter is chosen as ".". Therefore a three new fields will be created Day containing dd, Month containing MM and Year containing yyyy. The option "Length" is set to 2 for the day and month fields to keep the leading zeros if a values is lower than 10.

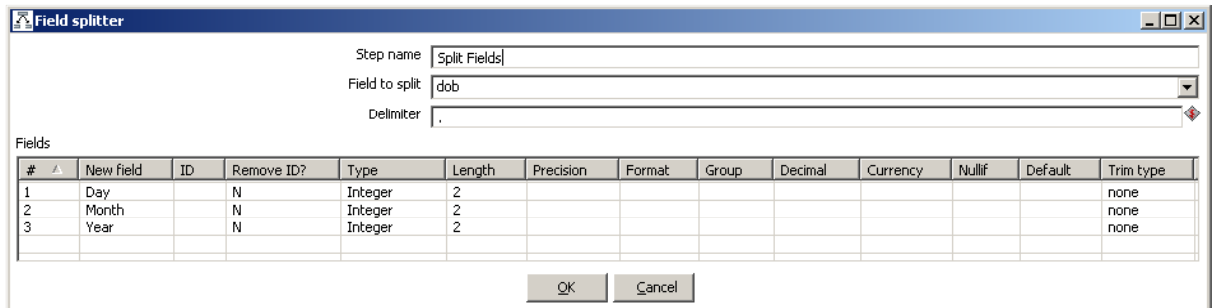


Figure 3-40 Splitting the dob field into three

In Figure 3-41 a preview of the Excel input with split Date of Birth field is shown.

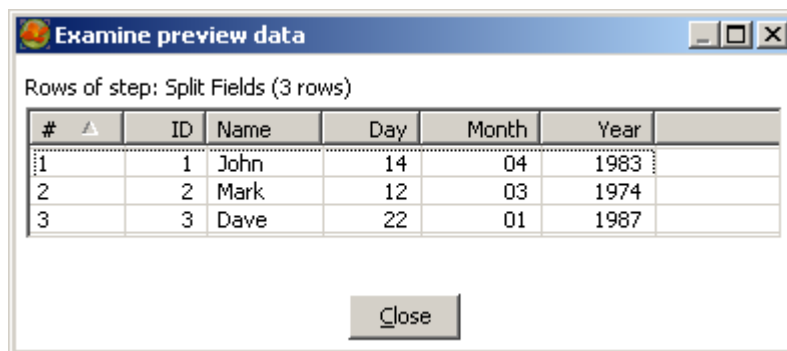


Figure 3-41 Preview of split Date of Birth field

3.3.9 Unique rows

This step deletes multiple occurrences of rows. The fields to compare can be listed. For instance in the example below (Figure 3-42) rows will be unique by "keynr". That means if "keynr" occurs multiple times in the input stream only the first will be added to the output stream, no matter if other fields are unequal or equal. If no entry is made the complete row is compared, which means every single field of a row must match to be deleted.

It is important that the rows are sorted before applying this step, using "Sort rows" (section 3.3.7). If this is not done only succeeding entries are removed if they match the set requirements.

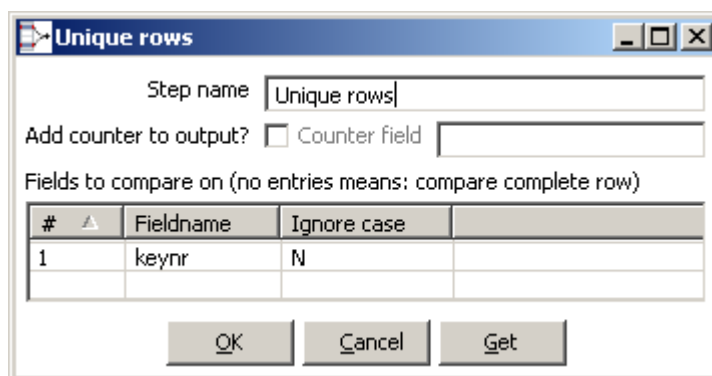


Figure 3-42 Unique rows properties

3.3.10 XSL Transformation

This step uses a xsl file to transform a XML field. It is for instance very useful to structure the hierarchy of a XML.

Therefore the name of the XML field has to be entered and the resulting XML field. The xsl file which is used can either be determined by a path in a field coming from a previous step, or the path is entered in the "XSL Filename".

The resulting field contains the transformed XML data as a string.

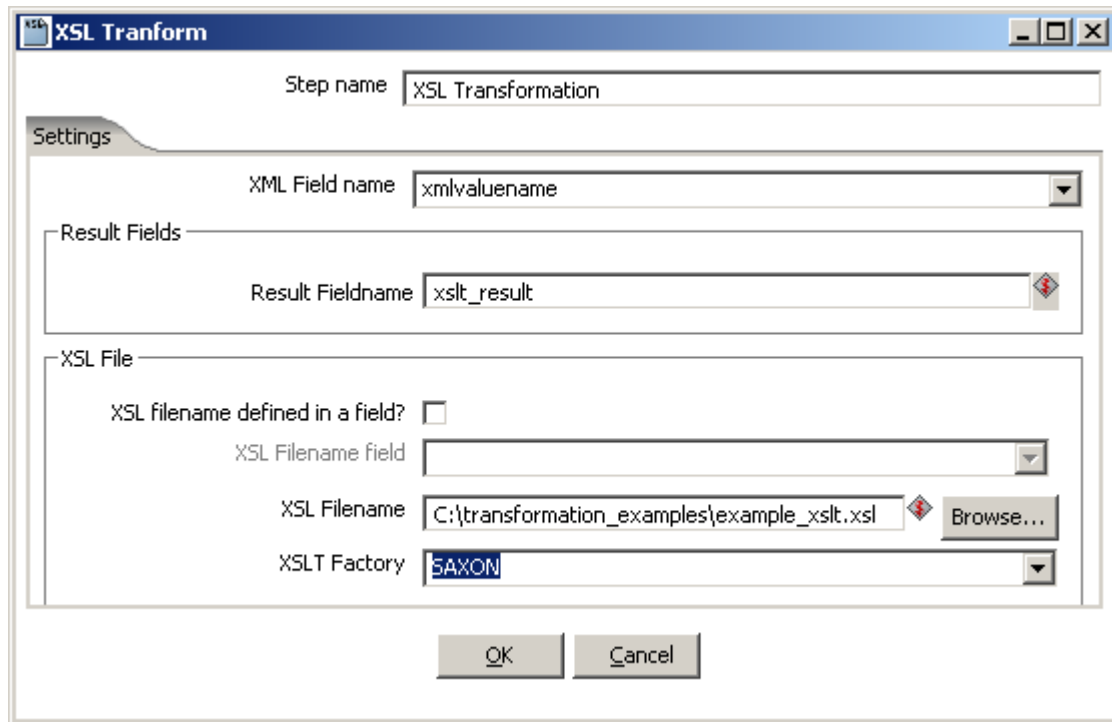


Figure 3-43 XSL Transformation properties

3.4 Scripting Steps

Scripting steps allow to include for instance Java scripts or SQL scripts in the transformation in order to modify data from a previous step.

3.4.1 Modified Java Script Value

In this step Java scripts can be added to modify data. The example below (Figure 3-44) shows a simple script to recombine the split Day of Birth again. The button "Get variables" makes variables in the script accessible as a field for following steps. With the "Test script" button, the current script can be run with test values.

In the folder "Transformation Functions" on the left many useful functions can be found to manipulate data. The input fields as shown in Figure 3-44 represent all fields coming from previous steps and can be used as variables in the script.

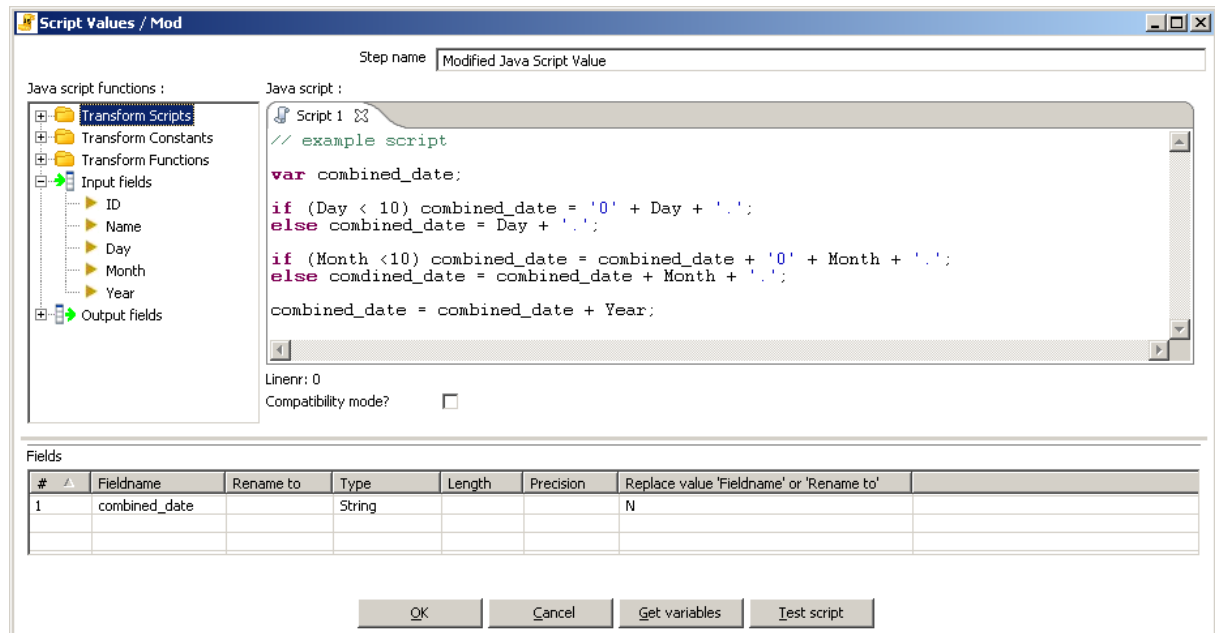


Figure 3-44 Script for combining Day, Month and Year again

3.5 Lookup Steps

The category “Lookup” offers for instance the HTTP client step to get information from a HTTP service.

3.5.1 HTTP client

The HTTP client step is used to retrieve information from a HTTP client. If it is not connected to a previous step that produces/sends rows, it needs to be triggered for example with a “Generate Rows” step (3.1.2).

The URL can be entered in the step, or taken from a field coming from a previous step. To choose the “Accept URL from field?” Check Box must either be checked or not.

The “Result filename” field contains the name of the output of this step, which is the information retrieved from a HTTP service. It can be used in following steps.

In the parameters list below a number of parameter-value pairs can be entered. The example in Figure 3-45 would produce a URL like `www.exampleurl.com/exampleservice.php?parameter1=value1¶meter2=value2¶meter3=value3`.

Note: The values are stored in fields. To produce the URL above value1 must contain the value “value1”, value2 must contain “value2” and so on. Constant values can not be entered right in this step. If a constant value is desired, it must be created with an “Add constants” step (3.3.2).

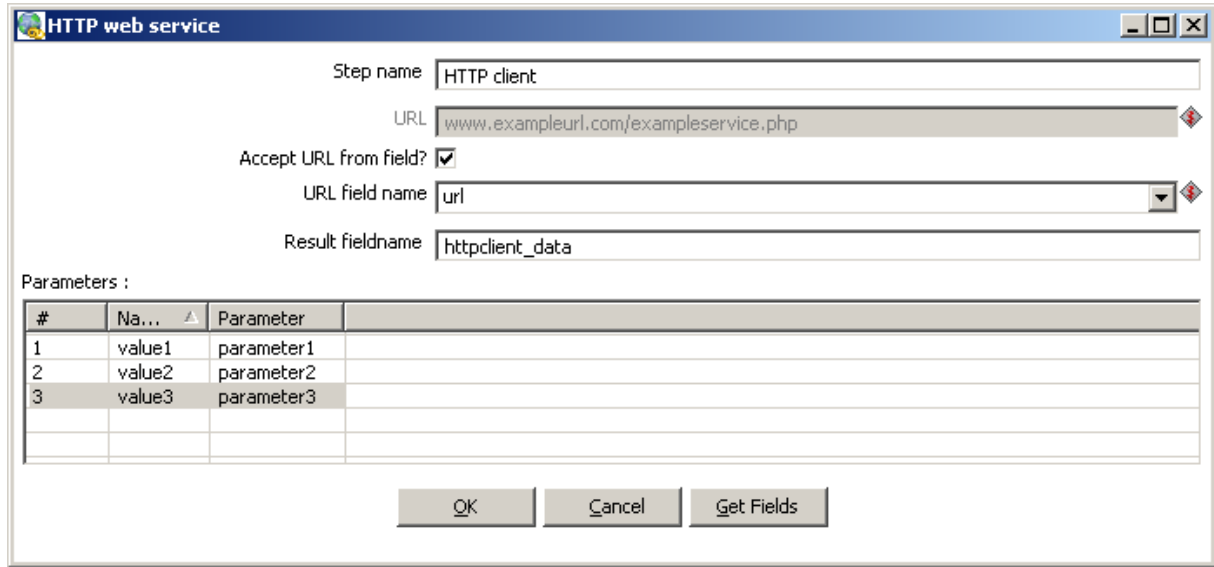


Figure 3-45 HTTP client properties

3.6 Join Steps

Join steps provide ways to join data coming from previous steps. For example selected values from a table or Excel input.

3.6.1 Merge Join

This step joins input data from previous steps over a certain key.

For instance the two Excel tables seen below need to be merged over ID.

	A	B	C
1	ID	Name	Date of Birth
2	1	John	14.04.83
3	2	Mark	12.03.74
4	3	Dave	22.01.87

Figure 3-46 First Excel Input

	A	B
1	ID	Nationality
2	2	US
3	3	GBR
4	1	AUS

Figure 3-47 Second Excel Input

It is very important that the tables are first sorted over the key to join them. Otherwise the result will not be correct. Then the "Merge Join" step can be applied and finally there is a "Select values" step in Figure 3-48 to just show Name, Nationality and Date of Birth of the result.

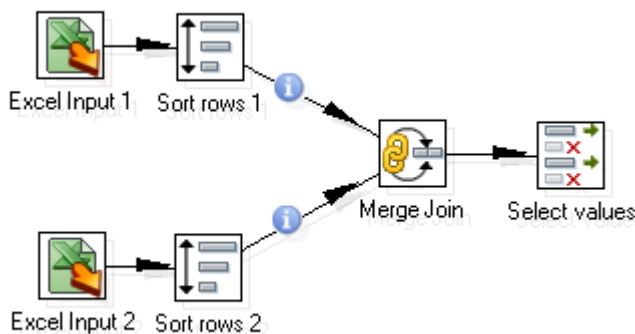


Figure 3-48 Transformation to join two Excel tables

In the configuration of the “Merge Join” step both input steps need to be entered. Then the join type can be selected. In this case “INNER” is chosen, because it is desired that only rows appear in the result, that have the same values for the key fields in both tables.

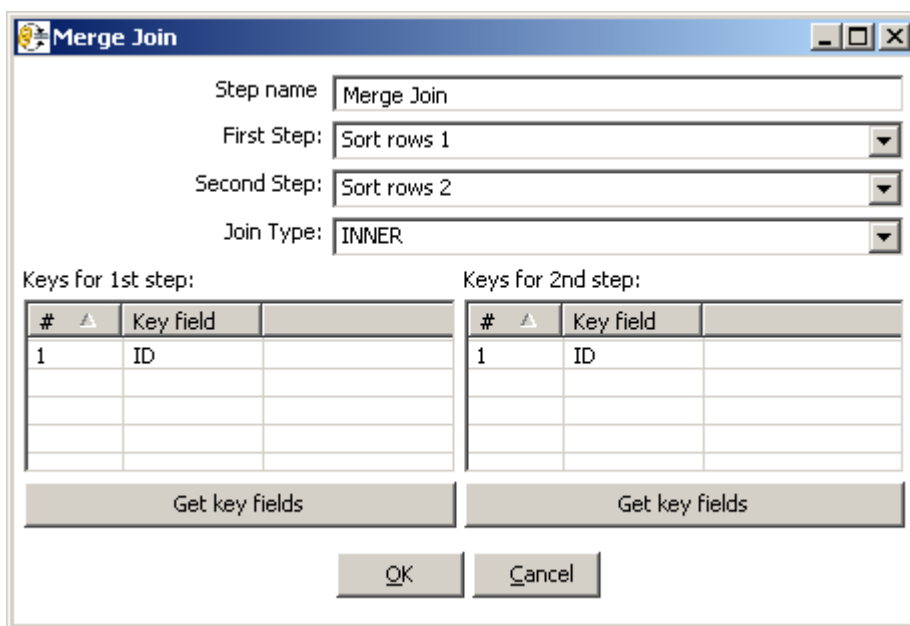


Figure 3-49 Configuration of the "Merge Join" step

The preview of this transformation can be seen in Figure 3-50.

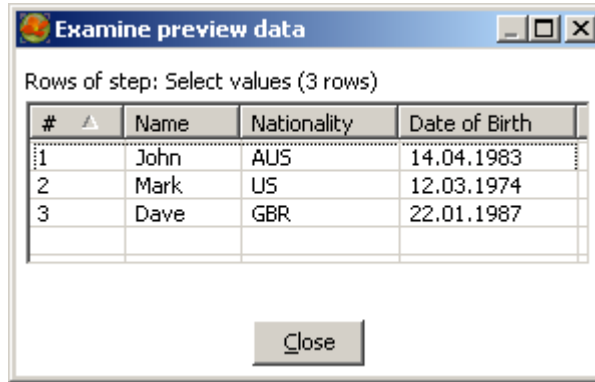


Figure 3-50 Preview of the join of two Excel tables

3.7 Job Steps

Basic jobs can be done like getting and setting variables. It is important to notice though, that in a transformation it is not possible to set a variable and use it afterwards. This is because all steps in Pentaho are basically executed in parallel.

Variables set before, either in a job or an other transformation called by a parent job can be used in transformations, for instance. It is also possible to use Pentaho's internal variables or system variables. They can be found, if "CTRL" + "SPACE" is pressed in a field where variables can be used (marked with a red \$ in a diamond).

3.7.1 Get Variables

In this step variables defined before (not in the same transformation!) can be stored as a field and used in the transformation. Simply a name and type for the resulting field has to be entered. The defined variable is has to be written with a leading "\$" and then in braces as can be seen in Figure 3-51.

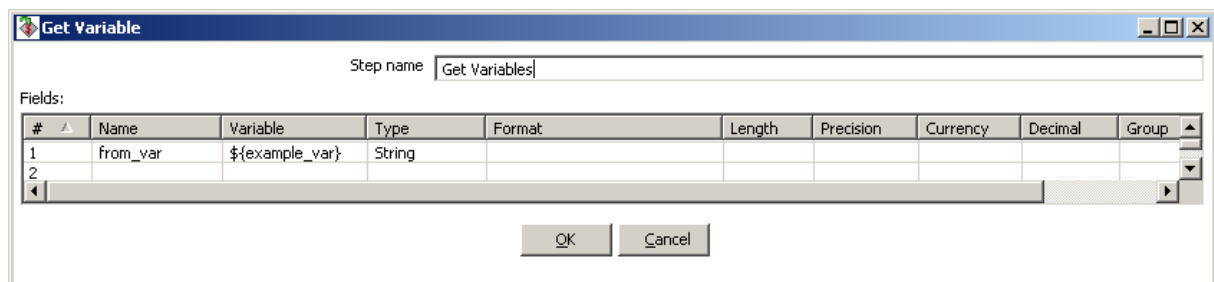


Figure 3-51 Get Variables step

3.7.2 Set Variables

Variables can also be created in a transformation. The "Set Variables" is used. Values coming from an input field are stored in a variable like in Figure 3-52.

The scope type is important. In the example “Valid in the root job” is selected. That means that the variable that is created can be accessed in the root job and in every other job and transformation below. If for example “Valid in the parent job” would be selected, the variable was not visible in a job calling the parent job of this transformation, but in all jobs or transformations called by the parent job of this transformation.

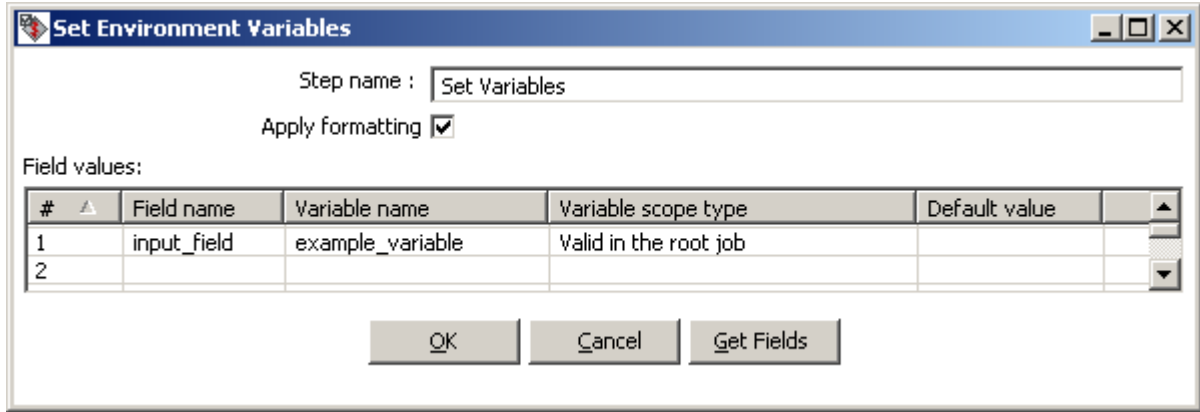


Figure 3-52 Set Variables

4 Pentaho Kettle Data Integration - Jobs

Here the most important job entries are explained and exemplified.

4.1 General entries

This category contains like starting the job or calling other transformations and jobs.

4.1.1 START

This entry is required for every job. It starts the job and offers some possibilities for scheduling. For instance a certain interval can be entered. Some other possibilities are to start the job daily or weekly, or enter no scheduling at all.

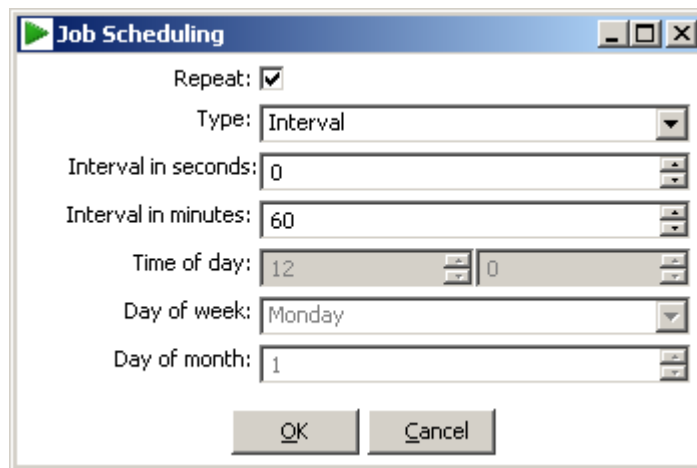


Figure 4-1 Properties of the START entry

4.1.2 DUMMY

The dummy job entry has no function at all, but it can be very useful to make the appearance of a job clearer, especially when loops are used.

The two figures below show the difference. Figure 4-2 performs exactly the same as Figure 4-3, but it is obviously not as clear to see. This can be much of a help especially in bigger jobs.



Figure 4-2 Loop without DUMMY

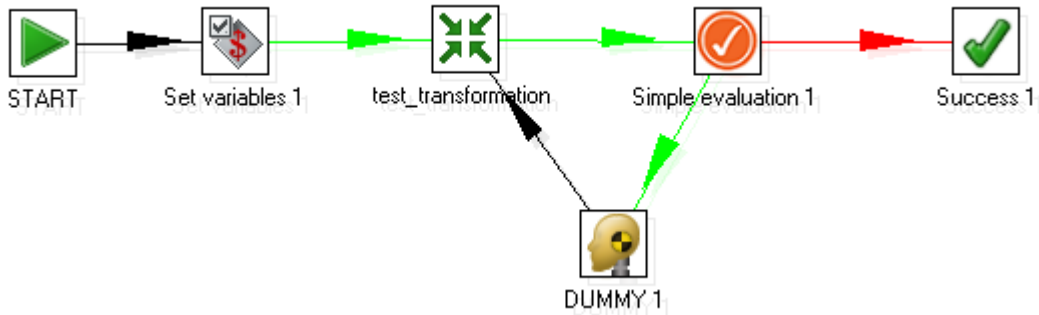


Figure 4-3 Loop with DUMMY

4.1.3 Abort Job

The current job is aborted and an error message can be shown. It is shown later in section 4 how it can be used in a job.

4.1.4 Display MsgBox Info

This step is configured very intuitively as seen in Figure 4-4 and creates a message box as in Figure 4-5. The title of the box can be entered as well as the message to be displayed.

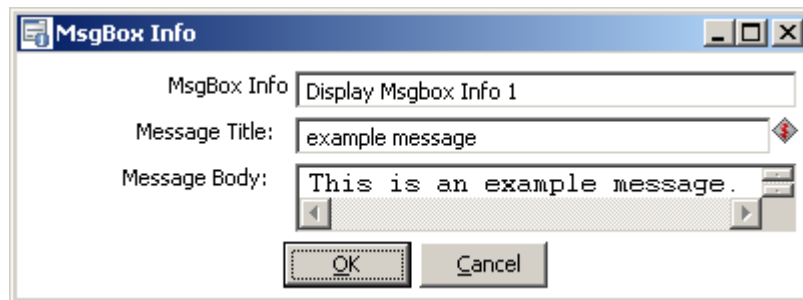


Figure 4-4 Creating a message box notification



Figure 4-5 Message box as created in Figure 4-4

4.1.5 Job

Inside a specific job with this entry other jobs can be called. To illustrate this two simple jobs are created. One creating a message box as in 4.1.4 and one just calling this job

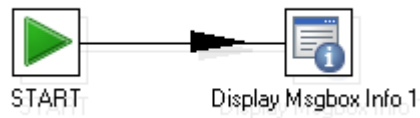
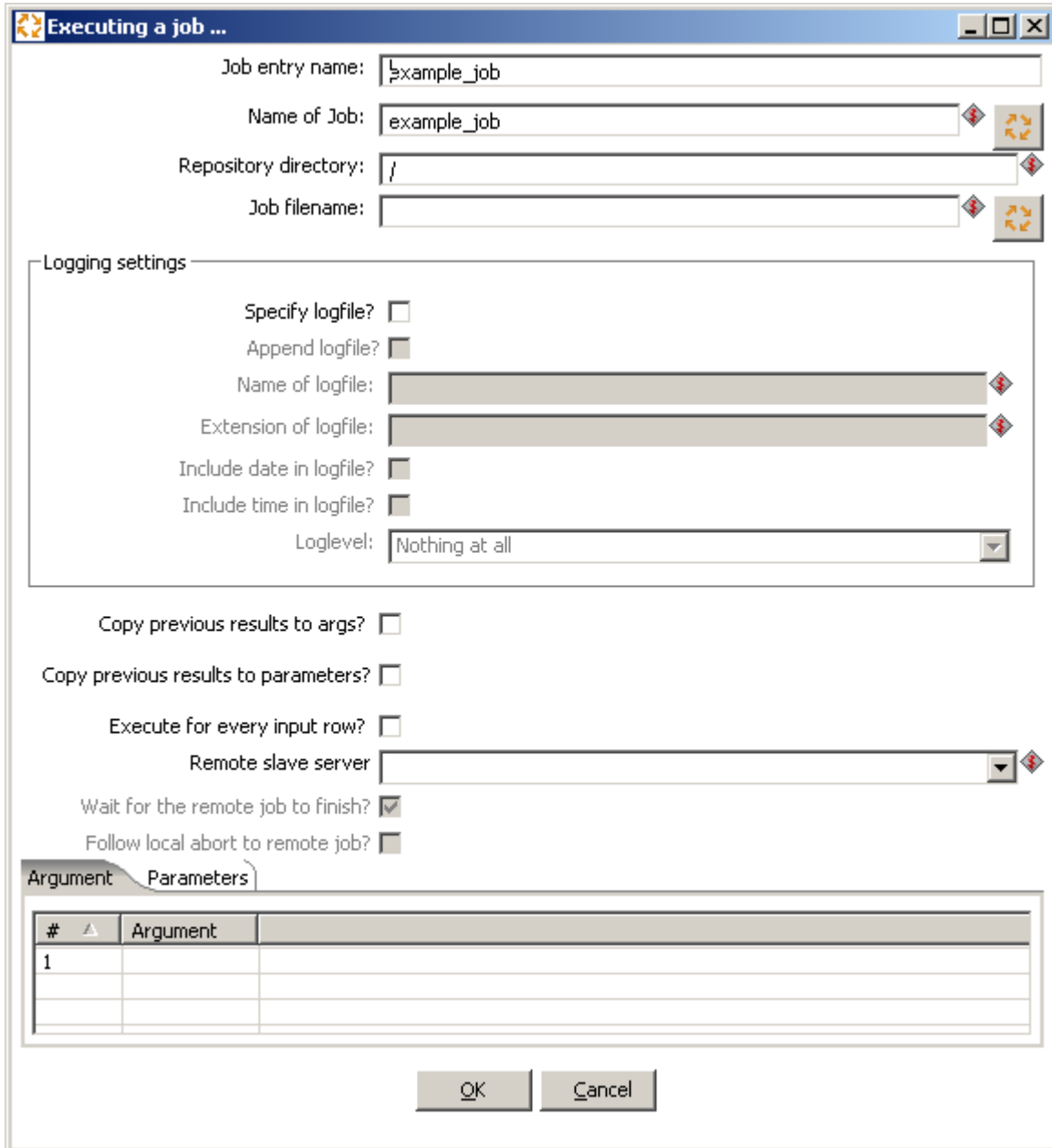


Figure 4-6 Example job creating a message box



Figure 4-7 Job calling the example job

In the "Name of Job" field the job to be executed can be selected. Basically this is sufficient information to start a job, but there are some options that can set.



Job entry name:

Name of Job:

Repository directory:

Job filename:

Logging settings

Specify logfile?

Append logfile?

Name of logfile:

Extension of logfile:

Include date in logfile?

Include time in logfile?

Loglevel:

Copy previous results to args?

Copy previous results to parameters?

Execute for every input row?

Remote slave server:

Wait for the remote job to finish?

Follow local abort to remote job?

Argument Parameters

#	Argument	Parameter
1		

OK Cancel

Figure 4-8 Properties for executing a job

4.1.6 Transformation

The same way as jobs can be called inside a job, transformations can also run. The properties window looks exactly like the one in Figure 4-8. The only difference is that a transformation is selected instead of a job.

An example of how a transformation can be integrated in a job is given at the end of section 4.

4.2 Mail entries

These entries offer various possibilities to work with e-Mails. Mails can be received or sent and addresses can be validated.

4.2.1 Mail

This entry can be used to send e-Mails. It can be useful for instance to report errors or simply finished jobs.

The tabs "Addresses" and "Servers" are equally to set up as in a usual e-Mail client and some contact details can be added like a phone number, if necessary.

In the group box "EMail Message" a few options can be selected. For instance to include the data in the sent message or setting the priority of the mail. Below the subject and a comment for a mail can be added.

It is also possible to attach files like a log file to the message. These options can be set in the "Attached Files" tab.

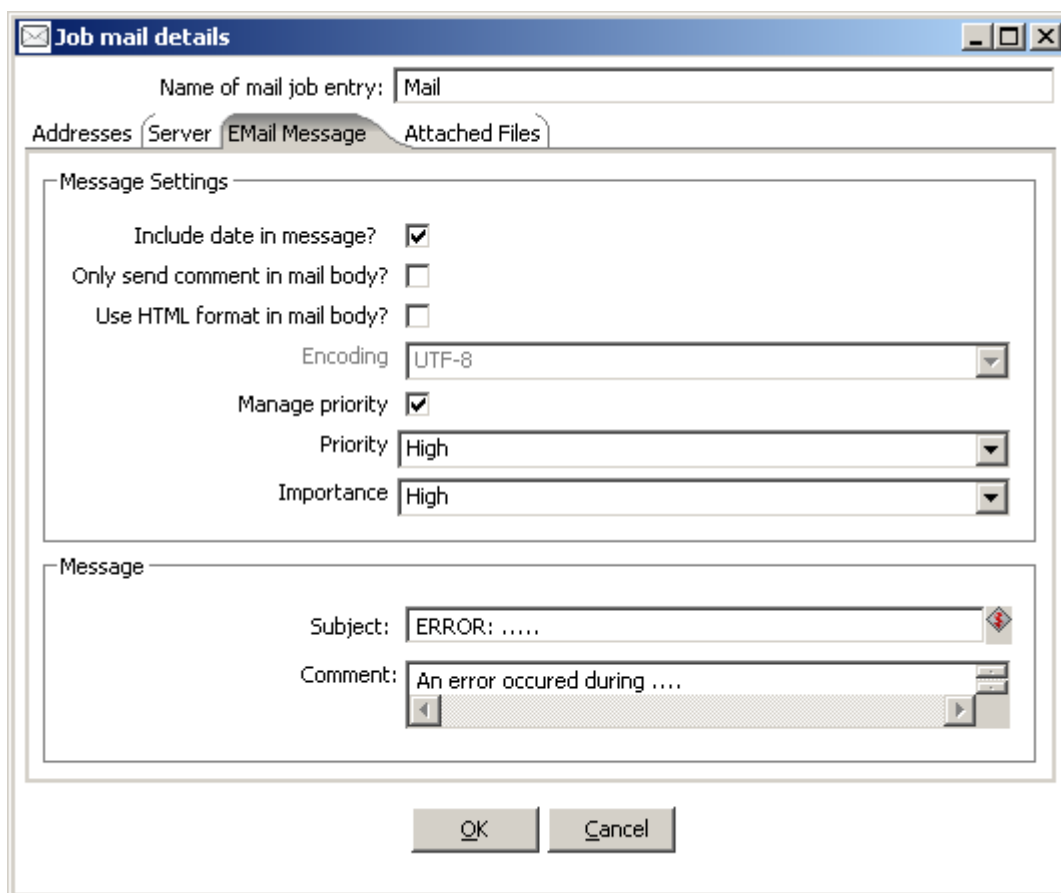


Figure 4-9 Properties for sending an e-Mail message

4.3 File management

With these entries some operations on files can be executed.

4.3.1 Create folder

A folder can be created in this entry. The folder name can be entered and variables can be used. If the "Fail if folder" Check Box is checked, this entry will fail if the folder already exists.

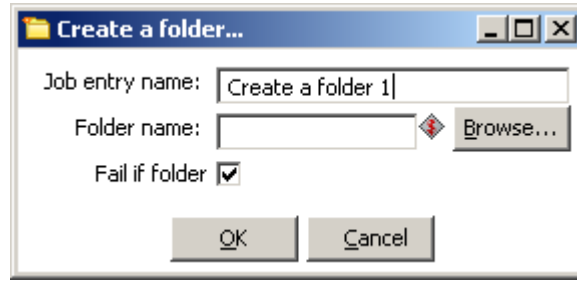


Figure 4-10 Create folder

4.3.2 Create file

This entry is very similar to the „Create folder“ entry (4.3.1) just with files.

4.3.3 Delete file

Files can be deleted, for instance if no longer needed after all transformations are finished.

4.3.4 Delete files

As the „Delete file“ (4.3.3) step just for delete multiple files. A folder can be selected and a RegEx can be entered, for instance to delete all files in this folder with the extension “xls”. If the “Include Subfolders” Check Box is checked files in subfolders are deleted as well. An example can be seen in Figure 4-11.

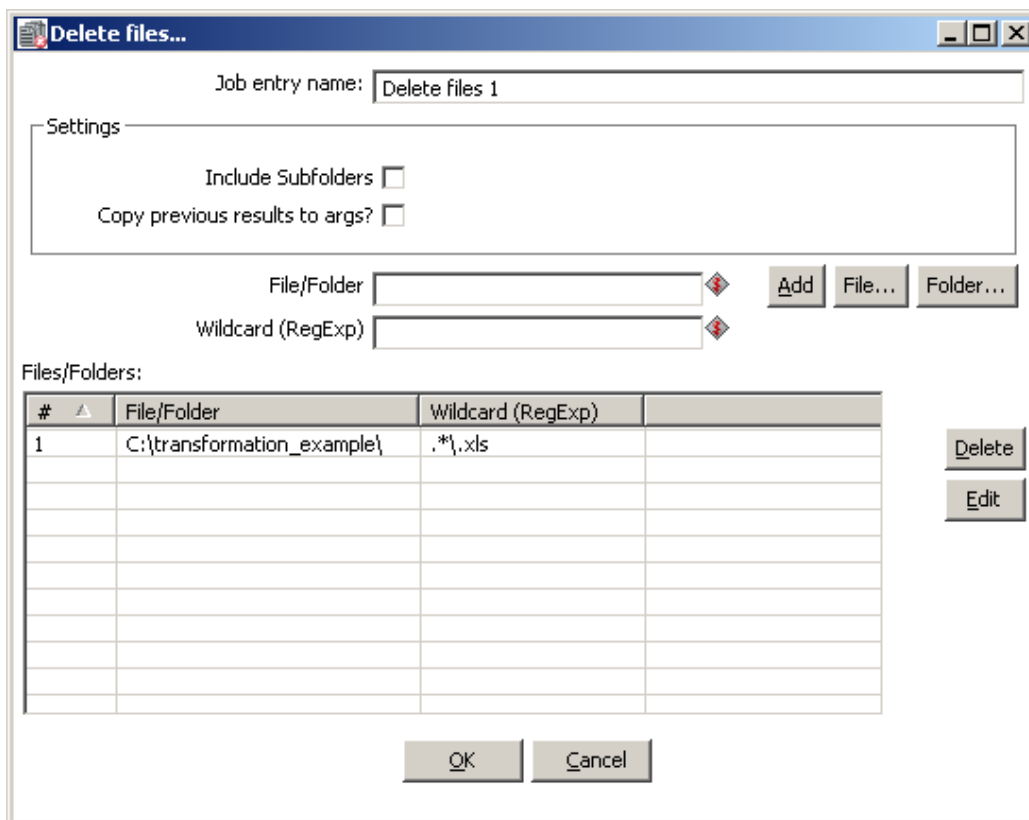


Figure 4-11 Delete files

4.3.5 Delete folders

This entry is very similar to the "Delete files" entry (4.3.4), but for deleting folders.

4.3.6 Move files

It can be very useful to move files at the end of a job. In this entry a source folder and a destination folder can be selected. If for instance the RegEx is entered as below all files are moved from "C:\transformation_examples" to "C:\transformation_examples\moved".

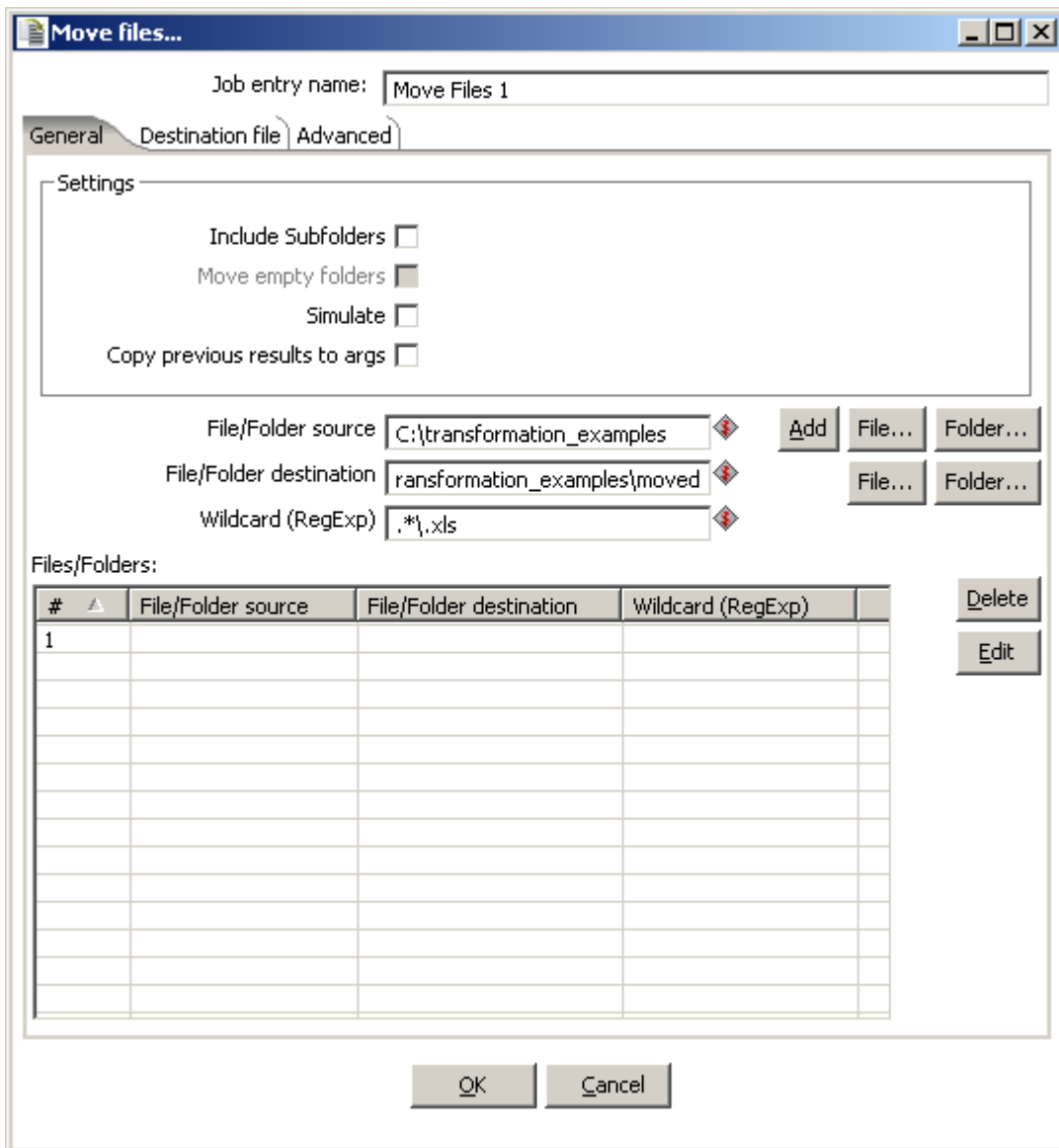


Figure 4-12 Move files

4.4 Conditions

Here conditional entries can be added in order to control the running job.

4.4.1 Simple Evaluation

Variables can be evaluated during a job. The example below (Figure 4-13) shows how the "Simple evaluation" entry can be used during a job.

A variable is set and then a transformation is called. In this transformation the variable can be manipulated. Afterwards it is evaluated. For instance as in Figure 4-14 it is checked, if the variable is smaller than 4. If this is true, the transformation is called again, otherwise the job is finished.

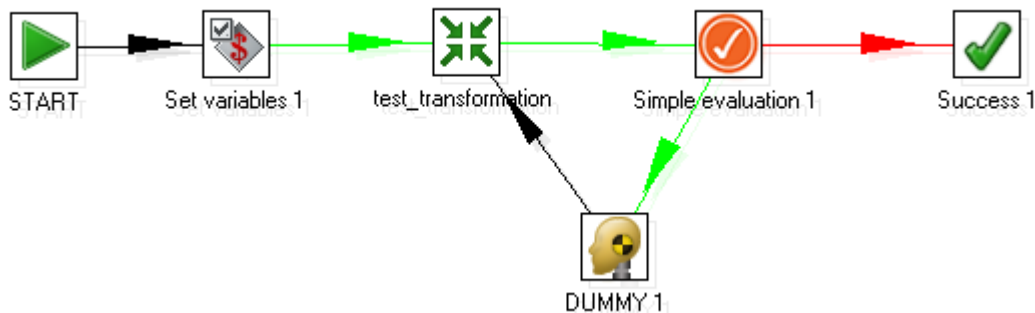


Figure 4-13 Simple evaluation in a job

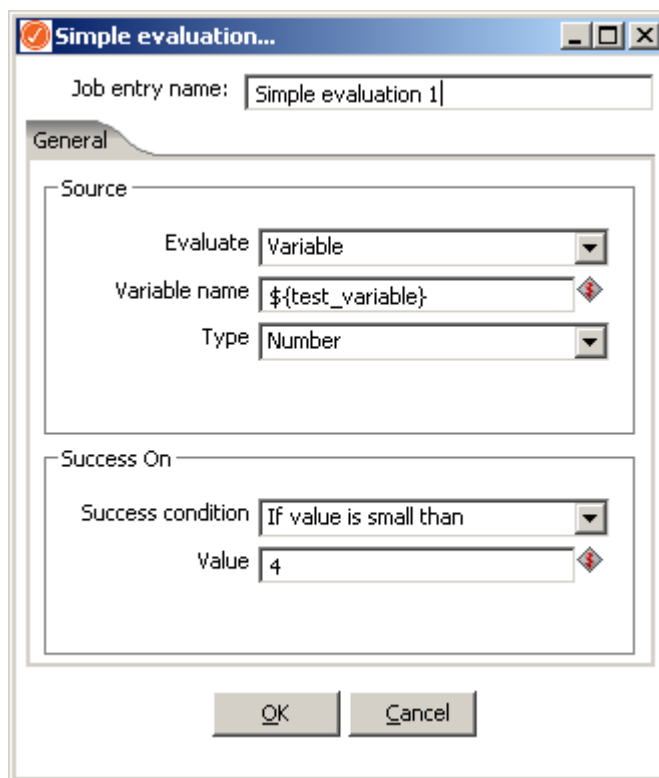


Figure 4-14 Simple evaluation properties

4.5 XML

These entries are handling XML files.

4.5.1

4.6 File Transfer

In this category several entries are provided to work with FTP and SFTP servers. The FTP, SFTP and SSH2 protocol can be chosen.

4.6.1 Get File with FTP

In the first tab the server information has to be entered as in Figure 4-15.

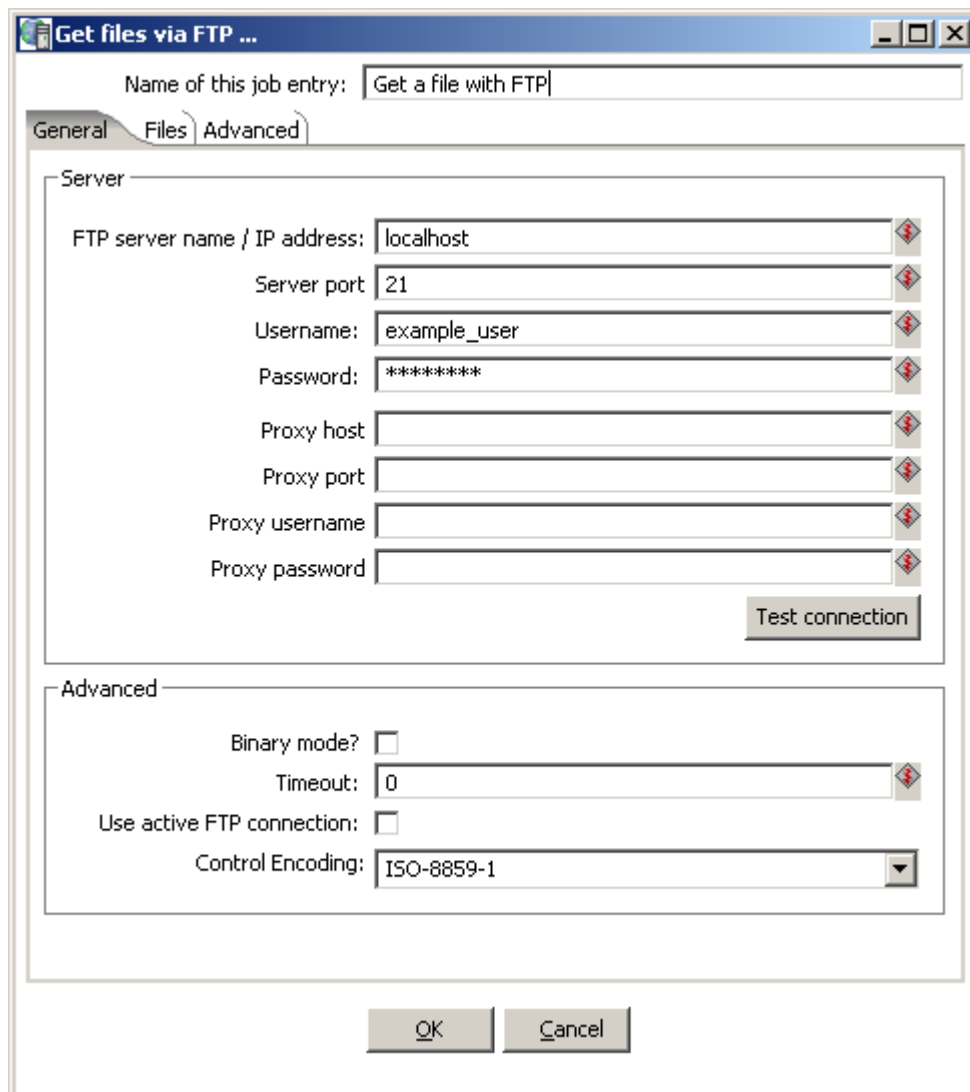


Figure 4-15 Entering server information

In the "Files" tab the remote directory has to be entered. In the field "Wildcard" either a filename can be entered or a RegEx as in Figure 4-16. This for example gets all files from the remote directory "getfrom" and copies them to the local directory "C:\transformation_examples", because this is entered below as target directory.

In the "Advanced" tab some conditions for success can be entered. For instance, that all has to work fine, or only a certain number of errors may occur.

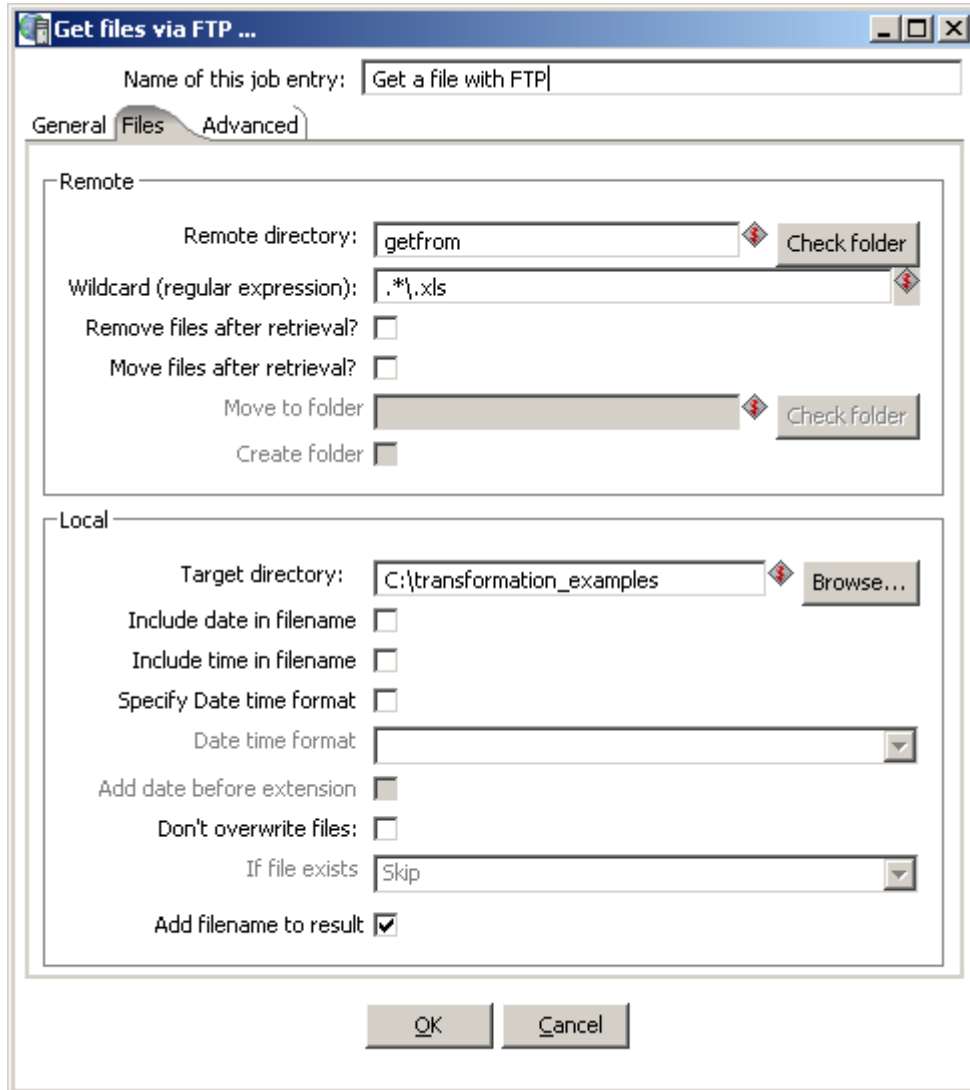


Figure 4-16 Setting file options

4.6.2 Put a file with FTP

As getting files from a FTP server it is also possible to put files from a local directory to an FTP server. The server information is entered equally as in Figure 4-15 in section 4.6.1.

In the "Files" tab the local directory and remote directory have to be entered. Again a file or RegExp can be specified. In Figure 4-17 all xls files from the local directory "C:\transformation_examples\out" are transferred to the remote directory "writeto".

The local files can be removed if the appropriate check box is checked.

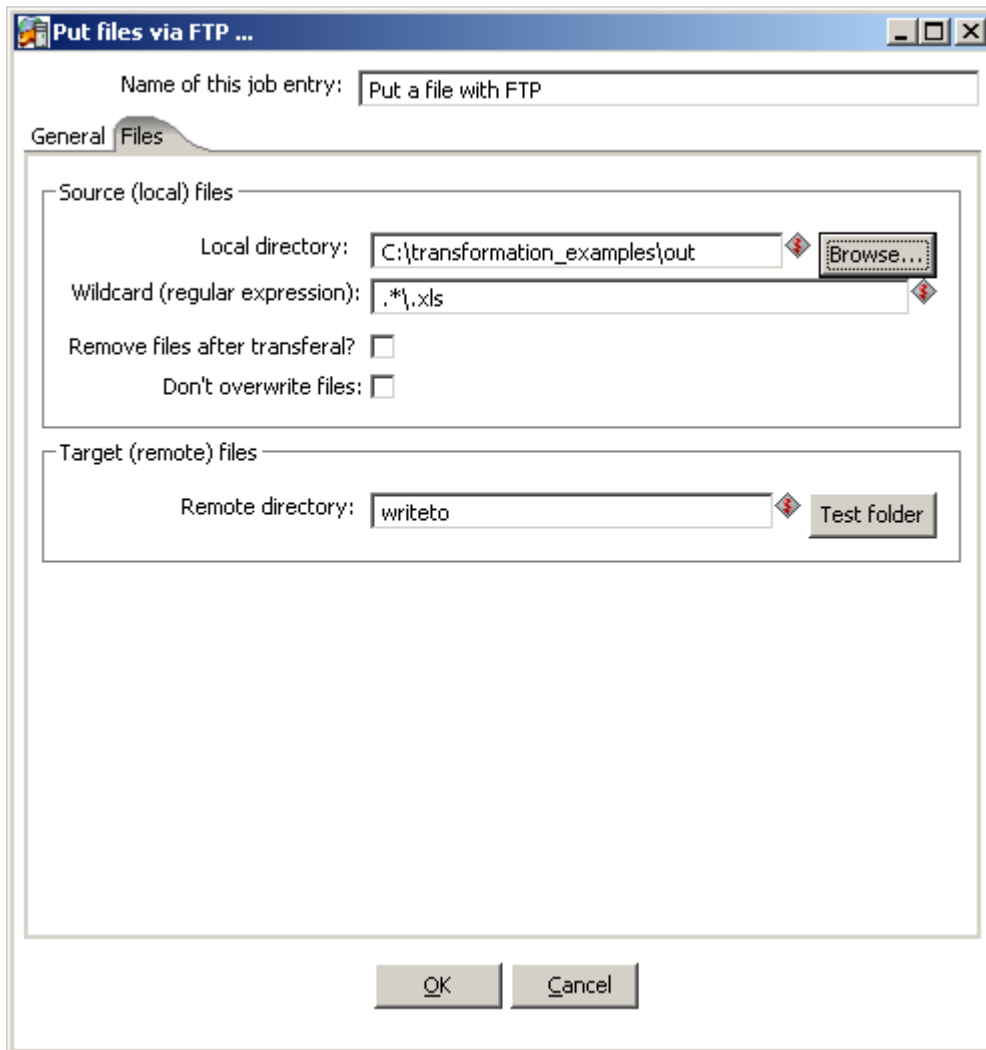


Figure 4-17 File options for putting files on a FTP server

4.7 Simple example how to combine some job entries and transformations

This example should show how jobs and transformation can be combined.

The intention is to retrieve one or more xls file(s) from a FTP server transform them in a certain manner and put the results back on the server again.

The two Excel sheets from Figure 3-46 and Figure 3-47 are stored in the remote directory "getfrom" on a FTP server. The transformation in Figure 3-48 should be applied, a result xls file created and this sent to the remote directory "writeto" on the FTP server again.

First an Excel Output step has to be added to the transformation of Figure 3-48 as described in section 3.2.1. The output file name will be "combined.xls" and is stored in "C:\transformation_example\out". It contains the three fields that were merged in the transformation.

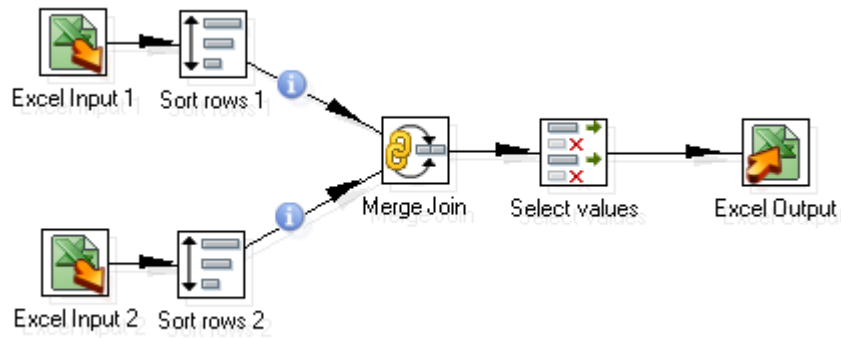


Figure 4-18 transformation to merge two excel sheets into one

Next the job is created to retrieve files, call the transformation and put the files back on the server. This can be seen in Figure 4-19. The abort job entries stop the execution of the job as soon there is an error in any step and displays an appropriate message. If the job is finished successfully a message box appears indicating that the files are now located on the FTP server.



Figure 4-19 Job to get files, transform them and put them back on a FTP server

The "Get file with FTP" step has to be configured as in section 4.6.1, because the remote directory is "getfrom" and the files are needed in "C:\transformation_example". The next entry is the transformation. Simply the sample transformation seen in Figure 4-18 has to be selected.

The "Put a file with FTP" step is configured that it puts all xls files from "C:\transformation_example\out" (this is where the transformation puts its results) to the remote directory "writeto" on the FTP server.

After executing the job this message box should appear.

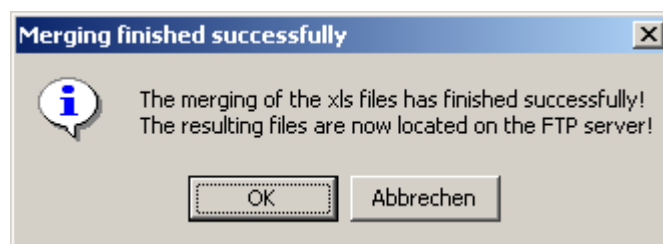


Figure 4-20 Message box after successful completion of the job

And in the remote directory "writeto" this Excel file can be found.

	A	B	C
1	Name	Nationality	Date of Birth
2	John	AUS	14.04.1983
3	Mark	US	12.03.1974
4	Dave	GBR	22.01.1987

Figure 4-21 Output Excel data on FTP server



5 Practical Examples

This section illustrates some practical examples using Pentaho Kettle Data Integration. The examples are shown and explained step by step.

5.1 ISPAN Data (DISMARC)

The ISPAN data is stored in an Excel file. Pentaho Data Integration is used to transform the input data according to the desired mapping and produce single XML files for every row of the input data.

The desired mapping is shown below. The root tag is called "regnet-document" with the attribute "version=1.0)", thus <regnet-document version="1.0">.

Table 5-1 shows how the data from the Excel sheets is mapped to XML files. In the column "XML tag name" the names of the XML tags in the final XML files are listed. If a tag has an attribute it is noted in the appropriate column. "XML parent section attribute" is the name of the attribute of the parent element <seccion> in which the current tag is located. The attribute names will either be "dmOAP" or "oaiInfo", because the basic structure of a final file looks like this:

```
<regnet-document version="1.0">
  <section name="dmOAP">
    <dmOAP-Identifier>Sygn: P0001A Pos: 03</dmOAP-Identifier>
    <dmOAP-Place role="recording">poznańskie</dmOAP-Place>
  </section>
  <section name="oaiInfo">
    <oai-Archive>ISPAN</oai-Archive>
  </section>
</regnet-document>
```

The excerpt above does only contain a few elements of the desired output.

In the column "Excel column name" the name of the field in the Excel data which is used for the current element is listed. If more fields are connected with a + more than one field is contained in this XML tag. If more than one fields are listed without pluses it means that each of these Excel columns produces an individual XML tag with the same name.

If a "*" follows the XML tag name, its content has special structured and is explained later in the document.

Those rows that are shaded brown-yellow get the information from a HTTP client. Their functions are explained later.

XML tag name	Attribute(s)	XML parent section attribute	Excel column name
dmOAP-Identifier *	-	dmOAP	"Sygn." + "Nr"
dmOAP-Title	-	dmOAP	Tytuł/Incipit tekstowy (gwara)
dmOAP-Subject	-	dmOAP	Wyk. 1: Rola, Wyk. 2: Rola Wyk. 3: Rola Wyk. 4: Rola Wyk. 5: Rola Wyk. 6: Rola Wyk. 7: Rola Wyk. 8: Rola
dmOAP-Subject-genre	encoding="dmGenres"	dmOAP	thesaurus
dmOAP-Description	-	dmOAP	Uwagi EsAC
dmOAP-Contributor *	role="performer"	dmOAP	"Wyk.1: Nazwisko, imię" + "Wyk. 1: Miejsce ur" "Wyk.2: Nazwisko, imię" + "Wyk. 2: Miejsce ur" "Wyk.3: Nazwisko, imię" + "Wyk. 3: Miejsce ur" "Wyk.4: Nazwisko, imię" + "Wyk. 4: Miejsce ur" "Wyk.5: Nazwisko, imię" + "Wyk. 5: Miejsce ur" "Wyk.6: Nazwisko, imię" + "Wyk. 6: Miejsce ur"

			<p>“Wyk.7: Nazwisko, imię” + “Wyk. 7: Miejsce ur”</p> <p>“Wyk.8: Nazwisko, imię” + “Wyk. 8: Miejsce ur”</p>
dmOAP-Contributor	Role=“creator”	dmOAP	Nagrał
dmOAP-Date-dateRecorded	-	dmOAP	Data nagr.
dmOAP-Date-dateRecorded	encoding=“dmEras”	dmOAP	thesaurus
dmOAP-Date-dateIssued	-	dmOAP	publikacja
dmOAP-Type	-	dmOAP	Type is a constant: “Sound”
dmOAP-Format	encoding=“dmFormats”	dmOAP	thesaurus
dmOAP-Language	-	dmOAP	Language is a constant: “POL”
dmOAP-Relation-hasSample	role=“audio”	dmOAP	service
dmOAP-Coverage-spatial	-	dmOAP	Reg.
dmOAP-Coverage-spatial	encoding=“dmGeography”	dmOAP	thesaurus
dmOAP-Place	-	dmOAP	Place is a constant: “Poland”
dmOAP-Place	role=“recording”	dmOAP	<p>Miejsce nagr.</p> <p>Gm.</p> <p>Pow.</p> <p>Woj.</p>
dmOAP-description	-	dmOAP	bibliografia
oai-Archive	-	oaiInfo	Archive is a constant: “ISPAN”
oai-Set	-	oaiInfo	Set is a constant:



			"ISPAN"
oai-InternalId *	-	oaiInfo	Sygn. + Nr
oai-DateStamp	-	oaiInfo	The date stamp contains the current date and time in this format: yy-MM-ddThh:mm:ssZ

Table 5-1 ISPAN mapping

Special structures in the output XML:

dmOAP-Identifier: This XML element contains data from two Excel fields, "Sygn." and "Nr". To mark the data constant text is added. The structure of the output looks like this: Sygn: "Sygn." Pos: "Nr", e.g. "**Sygn: P0001B Pos: 01**". P0001B and taken from the Excel table and "Sygn: " and " Pos: " is added as constants.

dmOAP-Contributor: Here two sources are used as well. "Wyk.1: Nazwisko, imię" and "Wyk. 1: Miejsce ur" for example. The structure is as follows: "Wyk.1: Nazwisko, imię" (place of birth: "Wyk. 1: Miejsce ur"), e.g. **Horemska Konstancja (place of birth: Trzcianka pow. Nowy Tomyśl)**. Horemska Konstancja is from field "Wyk.1: Nazwisko, imię" and **Trzcianka pow. Nowy Tomyśl** from field "Wyk. 1: Miejsce ur". The rest is added as constant text again.

oai-InternalId: The internal ID consists of "Sygn." and "Nr": ISPAN/00...00"Sygn." "Nr". For example the ID could look like this: **ISPAN/0000P0001B01**. "ISPAN/" is constant text. Then as many zeros are added as required to create a twelve digit number after the "/". For example if "Sygn." and "Nr" form **P0001B01**, four zeros are required to produce twelve digits.

5.1.1 Pentaho Data Integration Transformation

In this section it is explained how to create the transformation in Pentaho Data Integration.

In the new transformation (created by clicking File->New Transformation or by pressing CTRL-N) the Excel file(s) have to be imported. This required two input steps. The "Get File Names" and "Excel Input" steps. The "Get File Names" step passes the filenames to the next step. The directory where the "xls" files are located has to be added to the step as described in section 3.1.3. The regular expression is ".*\.xls" is entered to retrieve all files of type "xls".

In the "Excel Input" step check the "Accept file names from previous step" Check Box and chose the "Get File Names" step. As can be seen in Figure 5-14 the first step is renamed to "Get Excel Input", therefore in step "Excel Input" this has to be entered as the name of the previous step,



where the filenames are coming from. In the "Sheets" tab enter zeros for "Start row" and "Start column" and leave the "Sheet name" field empty. This will select all sheets from all Excel files imported. In the "Fields" tab click the "Get fields from header row ..." button and click OK. All fields will be sent to the next steps.

After this step the transformation splits up into two paths. One creating the dmOAP section and one creating the oaiInfo section of the final XML files. This could be done in one path of course, but it is clearer like this.

In the "Select values (OAP)" step all fields required for this section are chosen and renamed if needed. The list of values can be seen in Figure 5-1. These are all fields mentioned in Table 5-1. Those fields that are not changed anymore before being added to the XML can already be renamed to the right tag name. If there are fields that result in tags with equal names they can not yet be renamed to it, because fields with equal names could not be processed in the transformation steps. Field names that contain characters that are not suitable for JavaScript variable names for example should be renamed if it is possible that they have to be used in scripts.

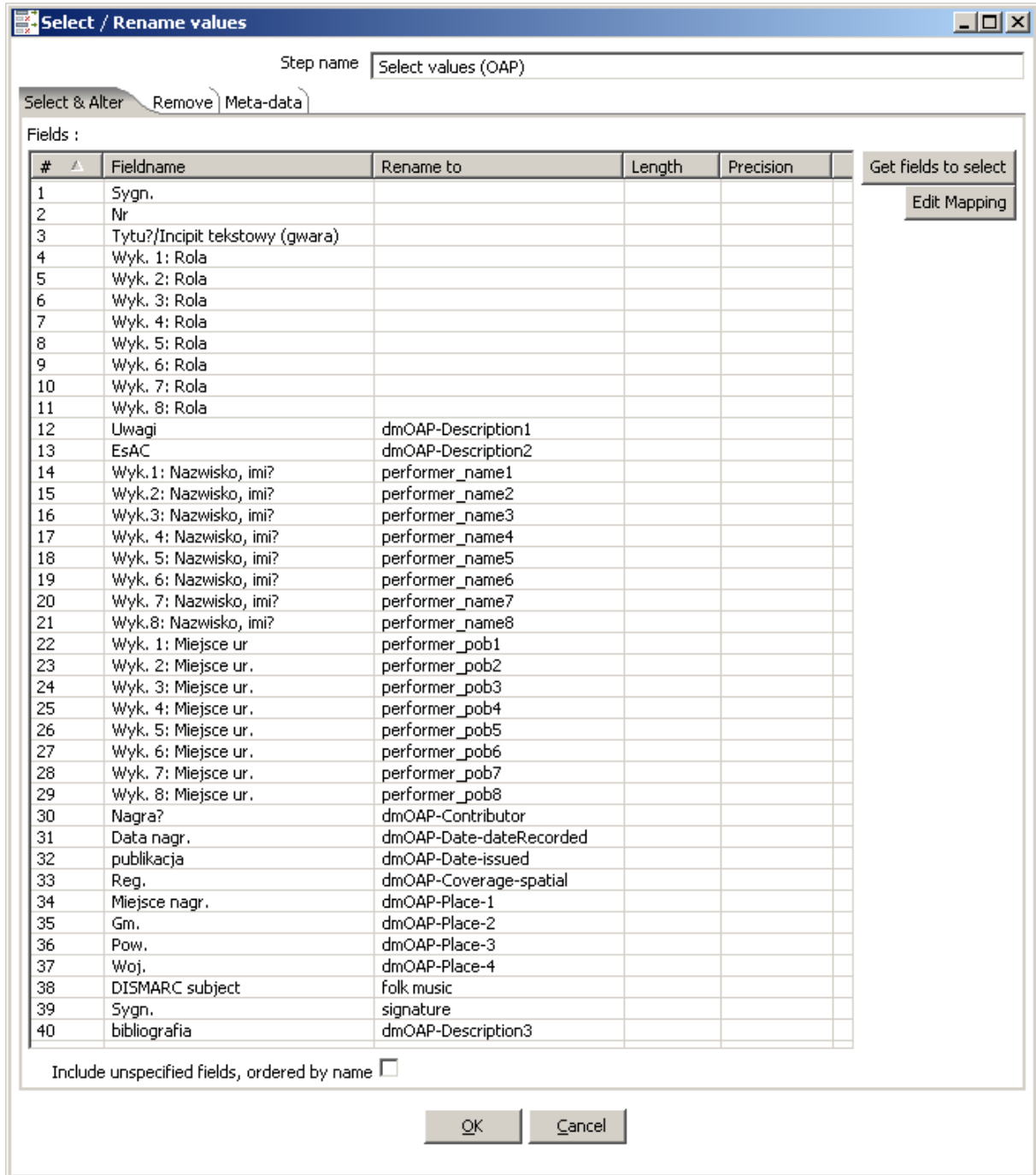


Figure 5-1 Selected fields for dmOAP section

In this step some constant values that are needed in the output are created. For instance is mentioned above the "Sygn: " and " Pos: " for the dmOAP-Identifier. In Figure 5-2 those constants can be seen. There are several "Add constant values" steps in this transformation, although all constants could be created in a single step. It is clearer though to use more steps and give these appropriate names, like the dmOAP attributes step which creates all constant values that are attributes for the XML elements (Figure 5-3).

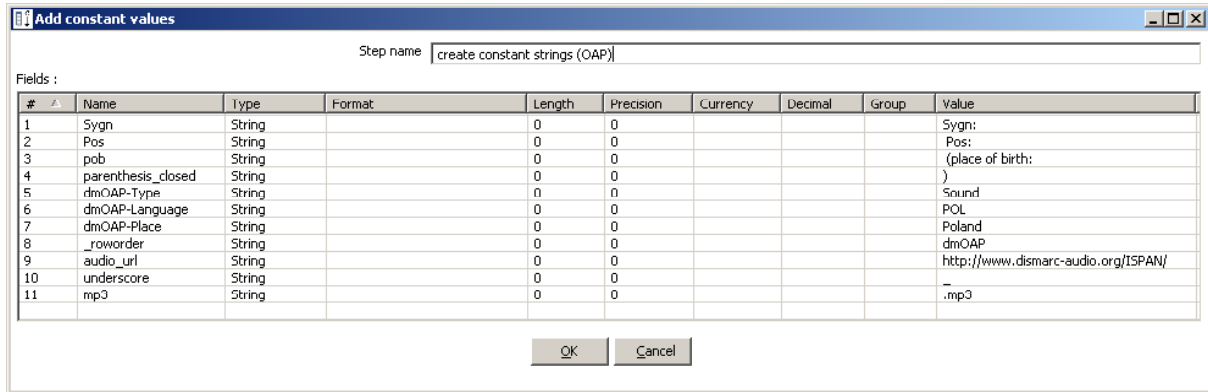


Figure 5-2 Required constant values

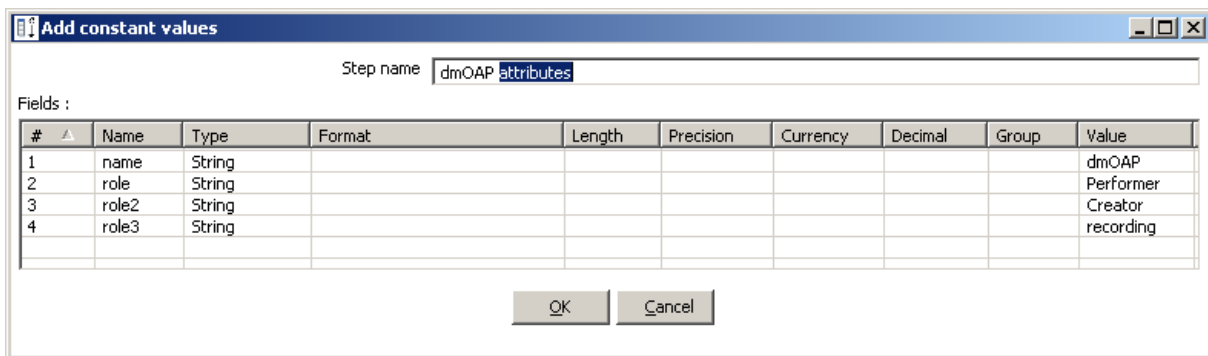


Figure 5-3 Add constant values - dmOAP attributes

The "merge strings" step produces the dmOAP-Identifier value as described above and the dmOAP-Contributor with role="Performer". Field values and constants are merged using the "A+B" and "A+B+C" calculations. This step also created the sample audio URL, which has the form [http://www.dismarc-audio.org/ISPAN/"Sygn."_"Nr".mp3](http://www.dismarc-audio.org/ISPAN/), e.g. http://www.dismarc-audio.org/ISPAN/P0202B_05.mp3.

Next are the HTTP client steps. These retrieve thesaurus information from a web-service as a string. Therefore several parameters are required. They will be added either as constants, if they do not depend on the input or be created for instance with a small JavaScript.

All web-services require the following parameters.

Parameter	Value
return	string
omitroot	1
encoding	cp1252
useentities	1

Table 5-2 parameters for all web-services

The URL is always the same <http://www.dismarc.org/services/rest/thesaurus.php> and therefore stored as a constant.

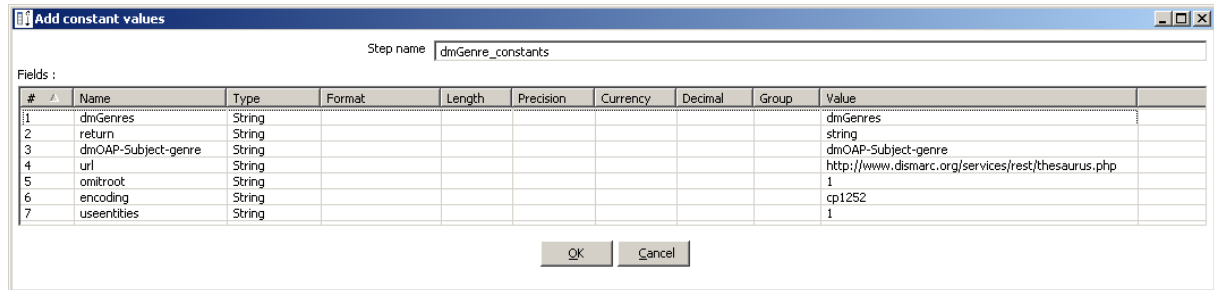


Figure 5-4 constants for web-services (dmGenre and general)

Besides, every web-service requires individual parameters. They are listed below.

- dmGenre:

Parameter	Value
thesaurus	dmGenres
term	folk music
fieldname	dmOAP-Subject-genre

Table 5-3 dmGenres parameters

Although folk music does never change for this mapping, it can be taken from the field "folk music". This was renamed from "DSIMARC subject" in the "Select values" step.

- dmFormats:

Parameter	Value
thesaurus	dmFormats
fieldname	dmOAP-Format
term	type of the record (Audio Tape or Gramophone record)

Table 5-4 dmFormats parameters

The value of the parameter term is not a constant this time. It is determined by a small java script. The information of the record type is held in the field "Sygn.". For use in a JavaScript it has to be renamed (e.g. to signature). The first letter of this fields values determines the record type. If it is a T the type is "Audio Tape" and if it is P the type is "Gramophone Record". The script simply stores the first letter of the signature in the variable firstChar using the function charAt(0). To define a

variable `var` has to be written in front of it. This happens in the second line when initializing the variable `record_type`. Note that it has no specific value assigned right now. Then there are two `if` statements checking if the letter is a T or a P. If a statement is true the command behind it is executed. This way `record_type` will be assigned the appropriate value depending on the first letter of the signature.

```
var firstChar = signature.charAt(0);
var record_type;

if (firstChar=='T') record_type = "Audio Tape";
if (firstChar=='P') record_type = "Gramophone Record";
```

The variable `record_type` has to be added to the fields list in the “determine record type” step to be able to use it as a parameter of the HTTP client.

- dmGeography:

Parameter	Value
thesaurus	dmGeography
term	Poland
fieldname	dmOAP-Coverage

Table 5-5 dmGeography parameters

- dmEras:

Parameter	Value
thesaurus	dmEras
term	expression depending on date recorded
fieldname	dmOAP-Date-dateRecorded

Table 5-6 dmEras parameters

The value of the parameter `term` depends on the date when the record was recorded. The value is determined with a JavaScript.

```
var date_year = dmEras_year;
var dmEras_term = getYearForDmEras(date_year);

function getYearForDmEras($date) {
    if($date == "19XX") {
        return "20th century CE";
    }

    if ($date == null) {
        return "";
    }
}
```

```
    }

    var jh = floor($date/100);

    if (jh == 19) {
        return trim($date.substr(0, 4) + "0s");
    }

    if ($date % 100 == 0) {
        jh--;
    }

    var cent = 0;
    var christ = "CE";
    if (jh >= 0) {
        cent = abs(jh) + 1 ;
        christ = "CE";
    } else {
        cent = abs(jh) ;
        christ = "BCE";
    }

    var num = "";
    switch (cent % 10) {
        case 1:
            num = "st";
            break;
        case 2:
            num = "nd";
            break;
        case 3:
            num = "rd";
            break;
        default:
            num = "th";
            break;
    }

    var term = cent + num + " century " + christ;
    return term;
}
```

After all required information for the dmOAP section has been gathered. The XML for this part can be created using the "Add XML" step. For the root element "section" has to be entered.

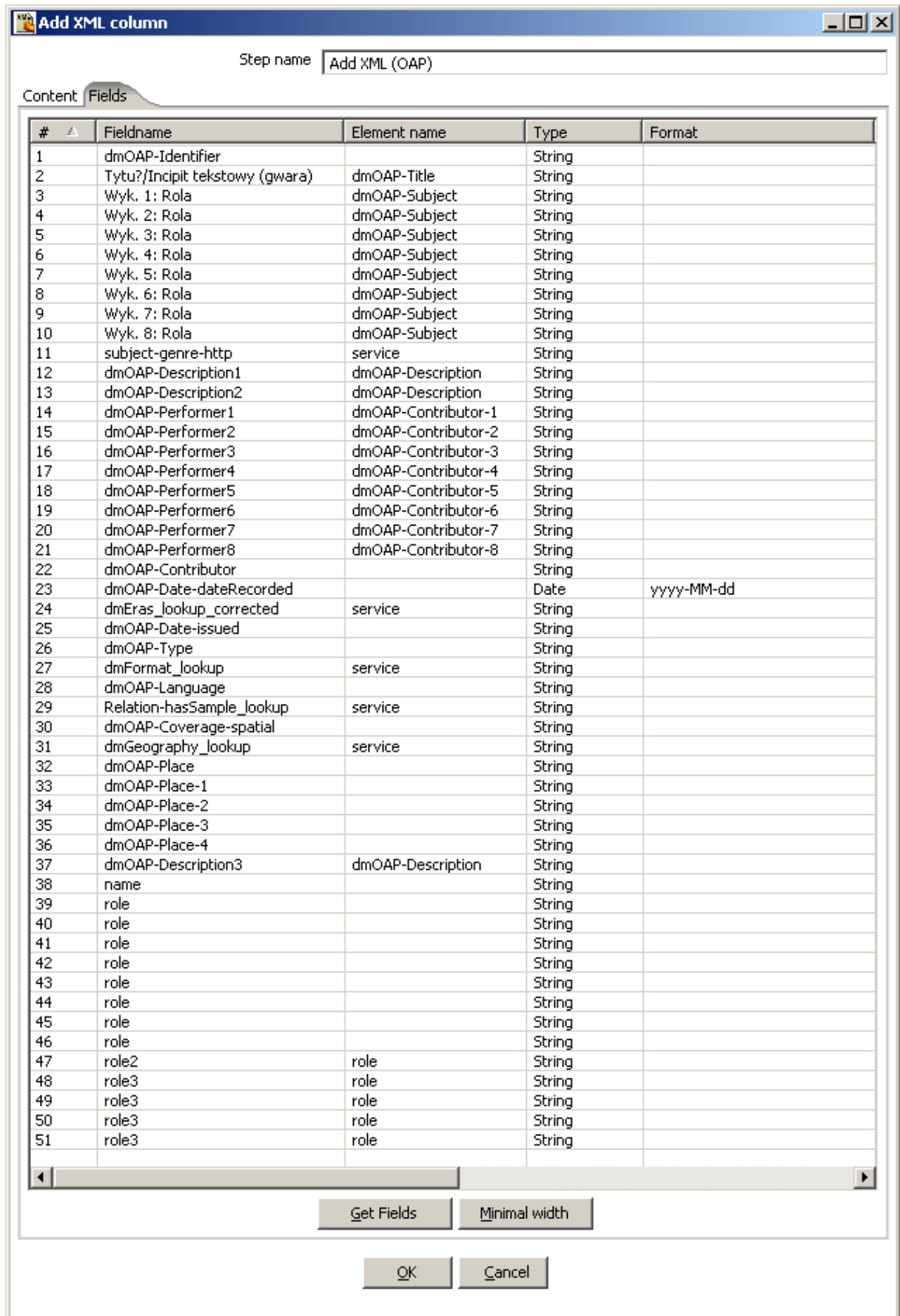


Figure 5-5 Add XML (OAP)

All entries in the list below (Figure 5-6) from entry number 38 are attributes of XML tags. In the column "Attribute" a "Y" has to be entered to mark them as attributes. "name" is the attribute of the section, that means the root element of the XML that is created in this step. Therefore the column "Attribute parent name" has to be left blank. "role" is the constant created before with value "performer" it is the attribute for the dmOAP-Contributor tag. It is not possible to assign an attribute to more elements, so for all eight dmOAP-Contributor-x the attribute has to be assigned. "role2" is has the value "creator" and is the attribute of the dmOAP-Contributor element. Later the dmOAP-Contributor-x entries will be renamed to dmOAP-Contributor , but therefore a style sheet transformation is required. The last attribute is role3 with value recording and is assigned to the dmOAP-Place-x elements. As above these elements will be renamed to dmOAP-Place without the numbers later.

35	dmOAP-Place-3		String			N	
36	dmOAP-Place-4		String			N	
37	dmOAP-Description3	dmOAP-Description	String			N	
38	name		String			Y	
39	role		String			Y	dmOAP-Contributor-1
40	role		String			Y	dmOAP-Contributor-2
41	role		String			Y	dmOAP-Contributor-3
42	role		String			Y	dmOAP-Contributor-4
43	role		String			Y	dmOAP-Contributor-5
44	role		String			Y	dmOAP-Contributor-6
45	role		String			Y	dmOAP-Contributor-7
46	role		String			Y	dmOAP-Contributor-8
47	role2	role	String			Y	dmOAP-Contributor
48	role3	role	String			Y	dmOAP-Place-1
49	role3	role	String			Y	dmOAP-Place-2
50	role3	role	String			Y	dmOAP-Place-3
51	role3	role	String			Y	dmOAP-Place-4

Figure 5-6 XML Add (OAP) attributes

After this step the XML still contains empty tags and those with the wrong names (dmOAP-Contributor1, dmOAP-Contributor2...). This can be corrected by using a style sheet transformation. Therefore a XSL Transform step has to be applied.

STYLESHEET DESCRIPTION

After the Style sheet was used the "Select for merging (OAP)" step is applied. It is needed, because the information from both paths have to be merged according to the values of "Sygn." and "Nr" in the Excel sheet. Therefore the XML field created in the previous step is selected and renamed to "row". The other fields required are "Sygn." and "Nr" for the sorting steps afterwards. Finally a constant is needed. "_roworder" has the value "dmOAP" and marks the name of the section.

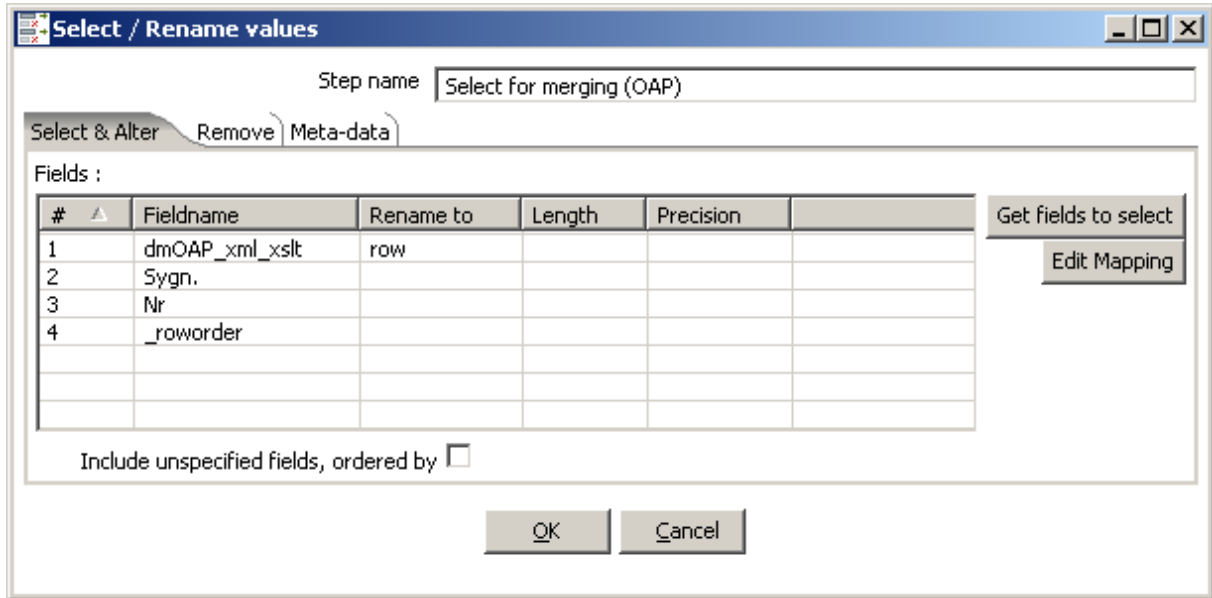


Figure 5-7 ISPAN transformation - Select for merging (OAP)

Now the first path of the transformation is ready to be merged.

The path for the "oaiInfo" section also starts with a select value step. The only fields that are needed from the Excel input are "Sygn." and "Nr", because they appear in the "internalID" tag of the output XML. In Figure 5-8 it can be seen, that those fields are selected twice. That is because for the JavaScript step creating the ID the original field names cannot be used, because they are not valid JavaScript variable names, so they are renamed to "identifier1" and "identifier2". Those not renamed are used later to merge both paths, the OAP and oai parts of the XML output.

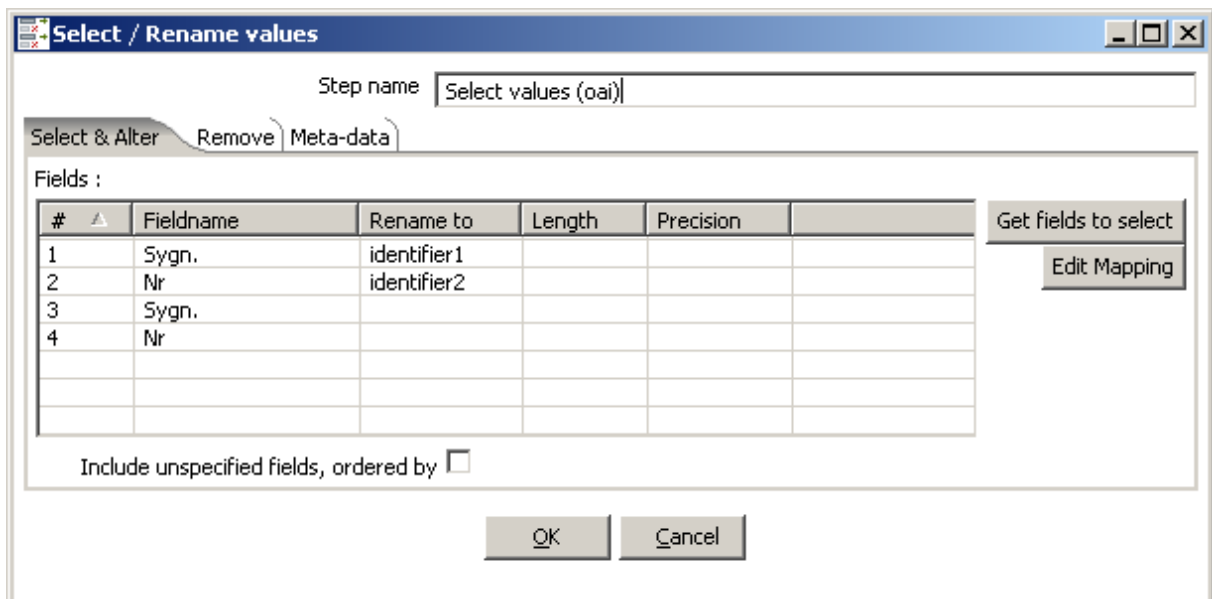


Figure 5-8 ISPAN transformation - Select values (oai)

Next constant strings are created. One called "ISPAN" with the same value "ISPAN" and a second one "_rowOrder" with the value "oaiInfo". The latter one is used for the merging process.

In the second "Add constants" step the name of the section is created that it can be set as an attribute of the section later. It has the same value as "_rowOrder", but it is clearer if this second constant is used.

Now the date stamp and the internal ID are created using JavaScript.

DESCRIPTION OF JAVA SCRIPT

After all information is available the XML can be created in the "Add XML (oai)" step. The constant "name" created before is set as an attribute of the root element (section). This can be done by selecting the field and enter a "Y" in the "Attribute" field and not entering a parent element. All other fields are given the correct element name as can be seen in Figure 5-9.

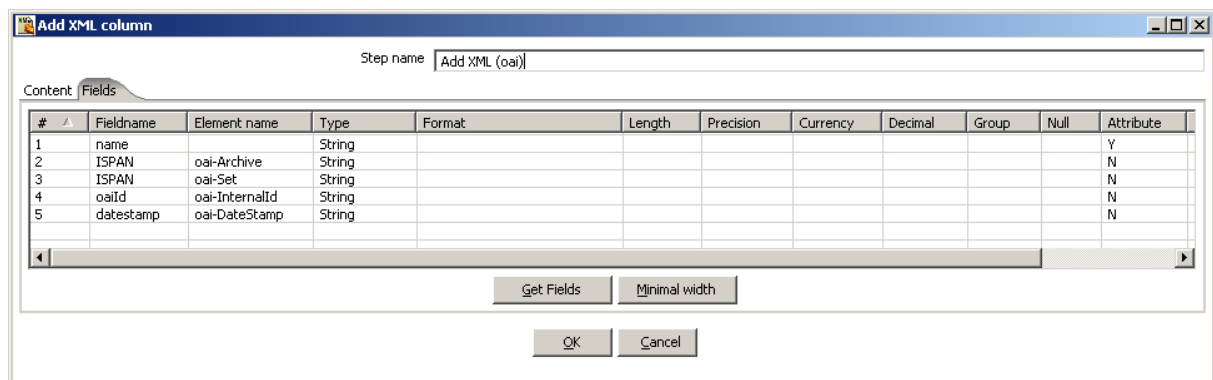


Figure 5-9 ISPAN transformation - Add XML (oai)

The select for merging step is similar to the one of the OAP path (Figure 5-7). The name of the XML field coming from the "Add XML field (oai)" has to be selected and "Sygn.", "Nr" and the "_rowOrder" constant.

Because it is necessary to get the right information from both parts into one XML file, a "Row flattener" step is used. Therefore all rows have to be sorted according to some specific keys first.

The fields are selected as seen in Figure 5-10. Both paths are then correctly ordered to flatten them, using the "Row flattener" step.

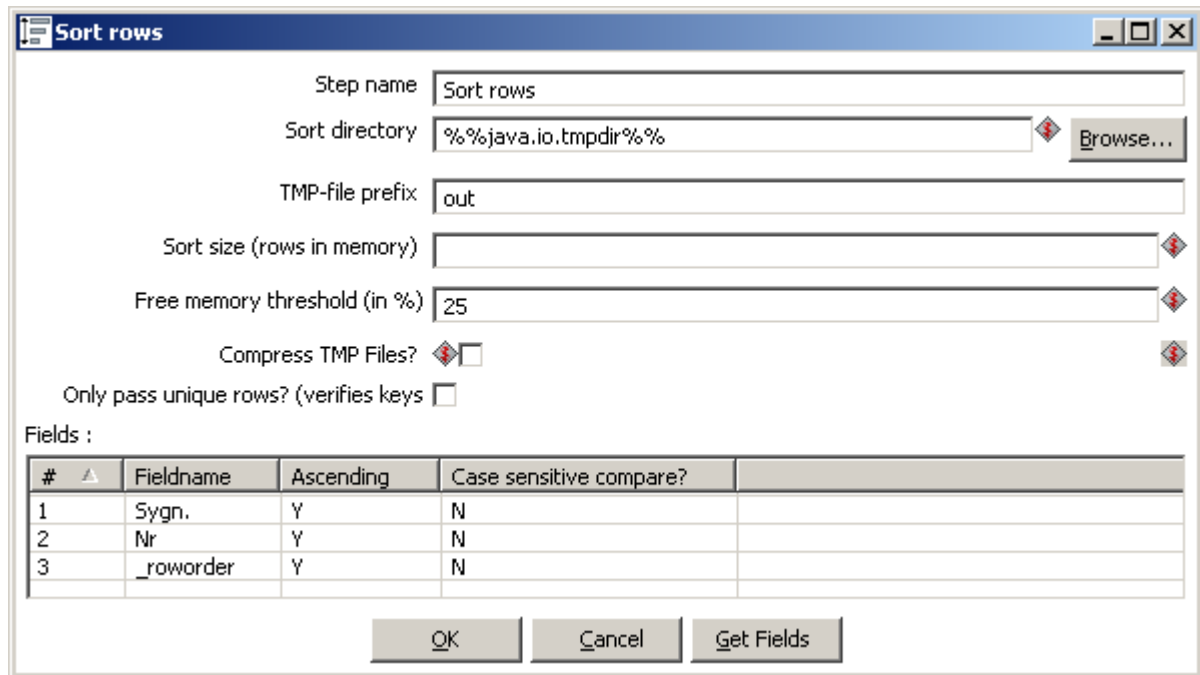


Figure 5-10 ISPAN transformation - Sort rows

First the fields for the "Row flattener" are selected. The only field to select is "row". As shown in Figure 5-11 this is the field that is flattened. Actually these fields are the "XML fields coming from the "Add XML" steps of both paths. In the "Select values" steps they were both renamed to "row" in order to be able to flatten them correctly.

The flattener produced two fields. The "oap" field containing the information of the "dmOAP" section of the final XML and the "oai" field for the "oaiInfo" section.

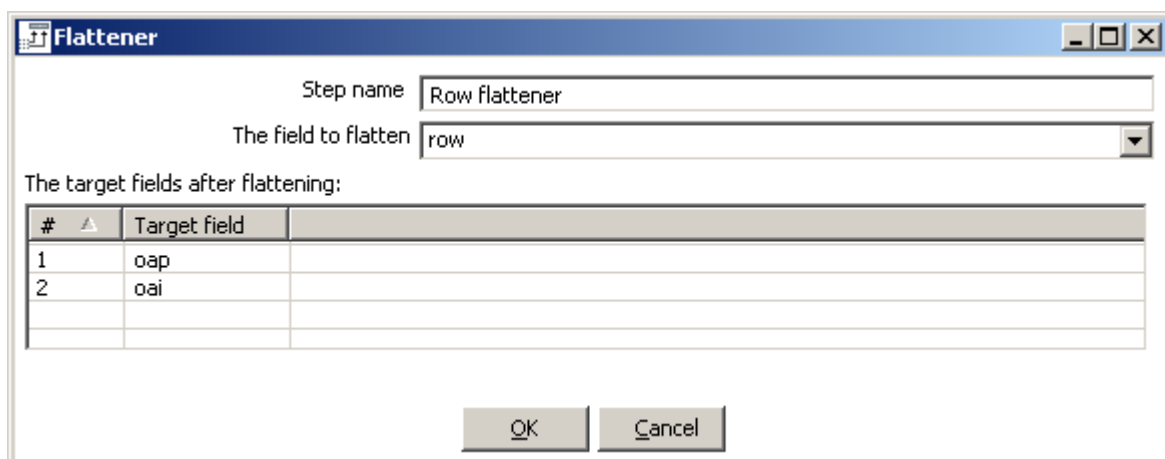


Figure 5-11 ISPAN transformation - Row flattener

In the following "Add constants" step the root element and the processing instruction of the XML file are created. The root element has the opening tag "<regnet-document version="1.0">" and the closing tag "</regnet-document>". The processing instruction looks like this: "<?xml

version="1.0" encoding="UTF-8"?>". Those constants are added to the XML file later in the "final output (xml)" step.

The last thing before the information can be written to a file is replacing or remove unwanted strings. Figure 5-12 below shows all necessary replacements.

"<" and ">" are HTML entities for "<" and ">". They have to be replaced, because the XML information from the HTTP lookups is stored as a string and original "<" and ">" are converted to those entities. If they are left in the file, they would not act as XML tags in the output file, but as string, which would result in wrong output. So the entities are simply replace by real "<" and ">".

The next thing is that all information from the web services are embedded in <service> tags. The are simply deleted, because it is not desired to have this information inside additional tags.

Another replacement is the <dm-OAP-Relation-hasSample> tag. It is deleted, if no related mp3 file exists and therefore the empty tag shown below is created. Due to this replacement no empty tag is in the output XML if no mp3 file exists.

Replacement number 6 is a regular expression. "<!--(.\s)*?-->" replaces all comments in the XML file. That can be useful if many were used during previous steps, but are not desired in the final XML file.

The last replacement is again replacing a HTML entity. "&" is the entity for the ampersand. So it is replace by the original "&" if such entities occur.

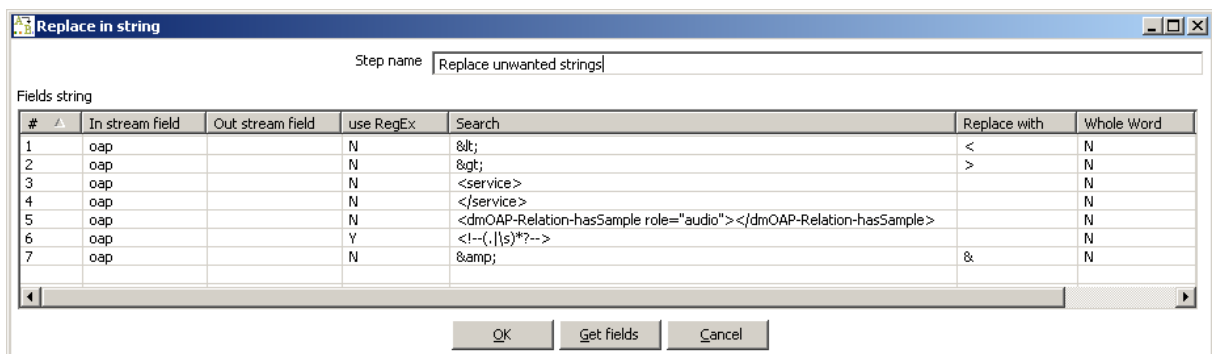


Figure 5-12 ISPAN transformation - replace unwanted strings

The last step is writing the fields into a XML file. Therefore a text file output step is selected. Then "Split every 1 row" needs to be entered in order to get a single XML file for every line in the Excel input.

The selected fields for the output can be seen in Figure 5-13. First the processing instruction and the intro are listed. These are constants for every row. Then the "oap" and "oai" sections are selected and finally the outro, which is the closing tag "</regnet-document>" is added.

Finally a XML file with the structure shown in section 5.1 is created.

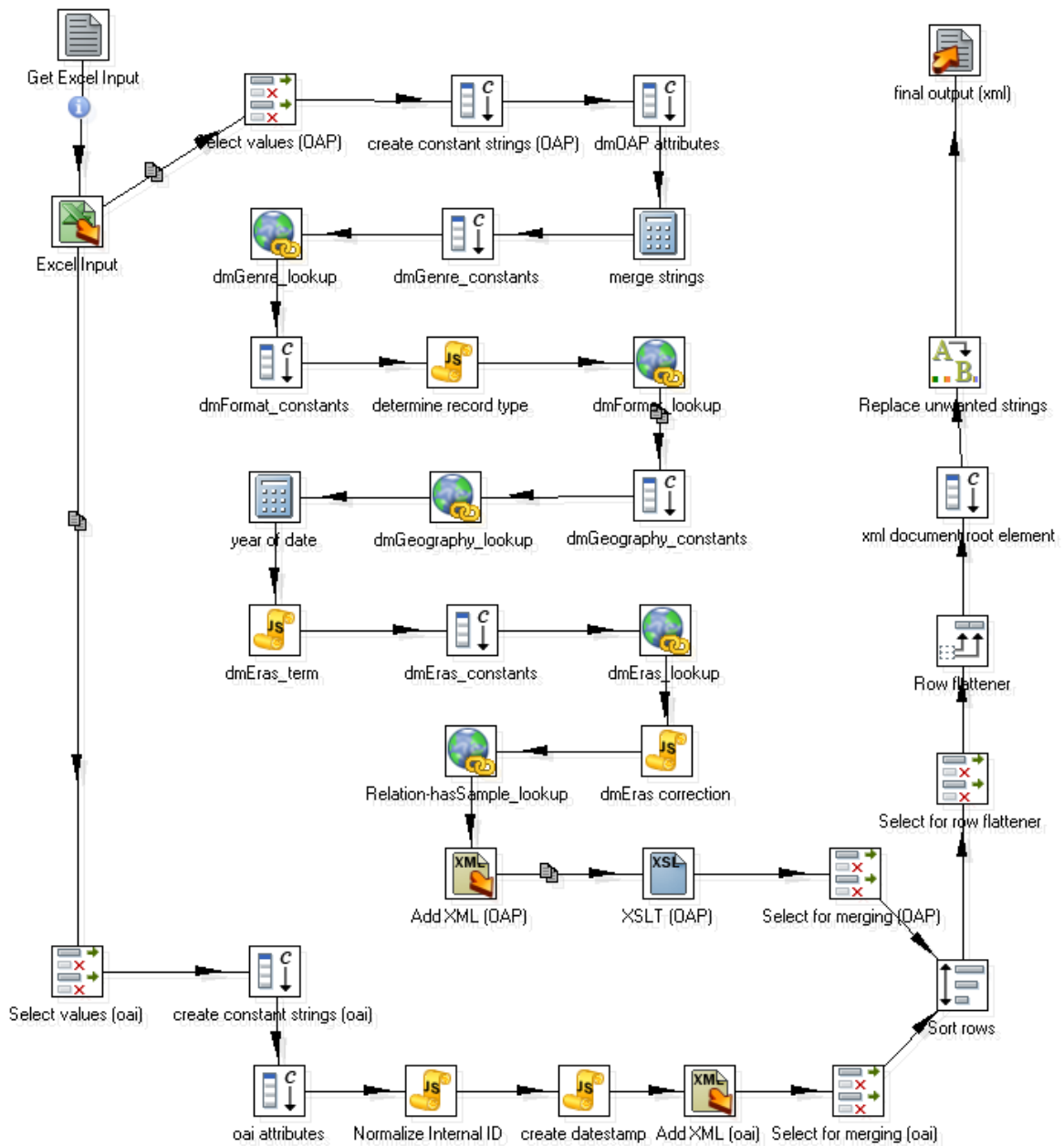


Figure 5-14 ISPAN transformation



Figure 5-15 ISPAN Job

5.2 BNF Data (BHL)

The BNF data comes in a XML file with the following structure:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <OAI-PMH>
  <responseDate>2009-07-20T14:04:20Z</responseDate>
  <request metadataPrefix="oai_dc" verb="ListRecords"
    set="gallica:5:56">http://oai.bnf.fr/oai2/OAIHandler</request>
- <ListRecords>
  - <record>
    + <header>
    + <metadata>
  </record>
  - <record>
    + <header>
    + <metadata>
  </record>
  - <record>
    + <header>
    + <metadata>
  </record>
```

Figure 5-16 BNF transformation - source XML structure

Every record has two sections <header> and <metadata>. The child tags of these elements contain the important information for every record.

```
- <record>
- <header>
  <identifier>oai:bnf.fr:gallica/ark:/12148/bpt6k97138g</identifier>
  <timestamp>2009-01-08</timestamp>
  <setSpec>gallica:5:56</setSpec>
  <setSpec>gallica:monographies</setSpec>
</header>
- <metadata>
- <oai_dc>
  <identifier>http://gallica2.bnf.fr/ark:/12148/bpt6k97138g</identifier>
  <title>Die fossile Flora der Polarländer. Mit einem Anhang über versteinerte Hölzer der
    arctischen Zone. Band 5 / von Dr. Oswald Heer,... ; von Dr. Carl Cramer,...</title>
  <creator>Heer, Oswald</creator>
  <publisher>IDC (Leiden)</publisher>
  <publisher>F. Schulthess (Zürich)</publisher>
  <date>1868-1883</date>
  <subject xml:lang="fre">Plantes -- Fossiles -- Régions polaires</subject>
  <description>Comprend : Einem Anhang über versteinerte Hölzer der arctischen
    Zone</description>
  <format>50 microfiches ; 105*148 mm</format>
  <language>ger</language>
  <relation>Notice du catalogue :
    http://catalogue.bnf.fr/ark:/12148/cb37273206g/description</relation>
  <type xml:lang="eng">text</type>
  <type xml:lang="fre">monographie imprimée</type>
  <type xml:lang="eng">printed monograph</type>
  <rights xml:lang="fre">domaine public</rights>
  <rights xml:lang="eng">public domain</rights>
  <format>application/pdf</format>
  <source>Bibliothèque nationale de France</source>
</oai_dc>
</metadata>
```

Figure 5-17 BNF transformation - header and metadata structure

In Figure 5-17 the names and structure of the elements can be seen. The desired output is a XML file in MODS schema.

Pentaho offers an input step called "Get data from XML" (3.1.4). By using XPath statements the values of the elements above can be extracted and transformed.

In the "Content" tab of the "Get data from XML" step a loop XPath element has to be entered. In this case it is "/OAI-PMH/ListRecords/record". That is because the element record is the repeating element in the source XML and the information for every single record is inside such an element.

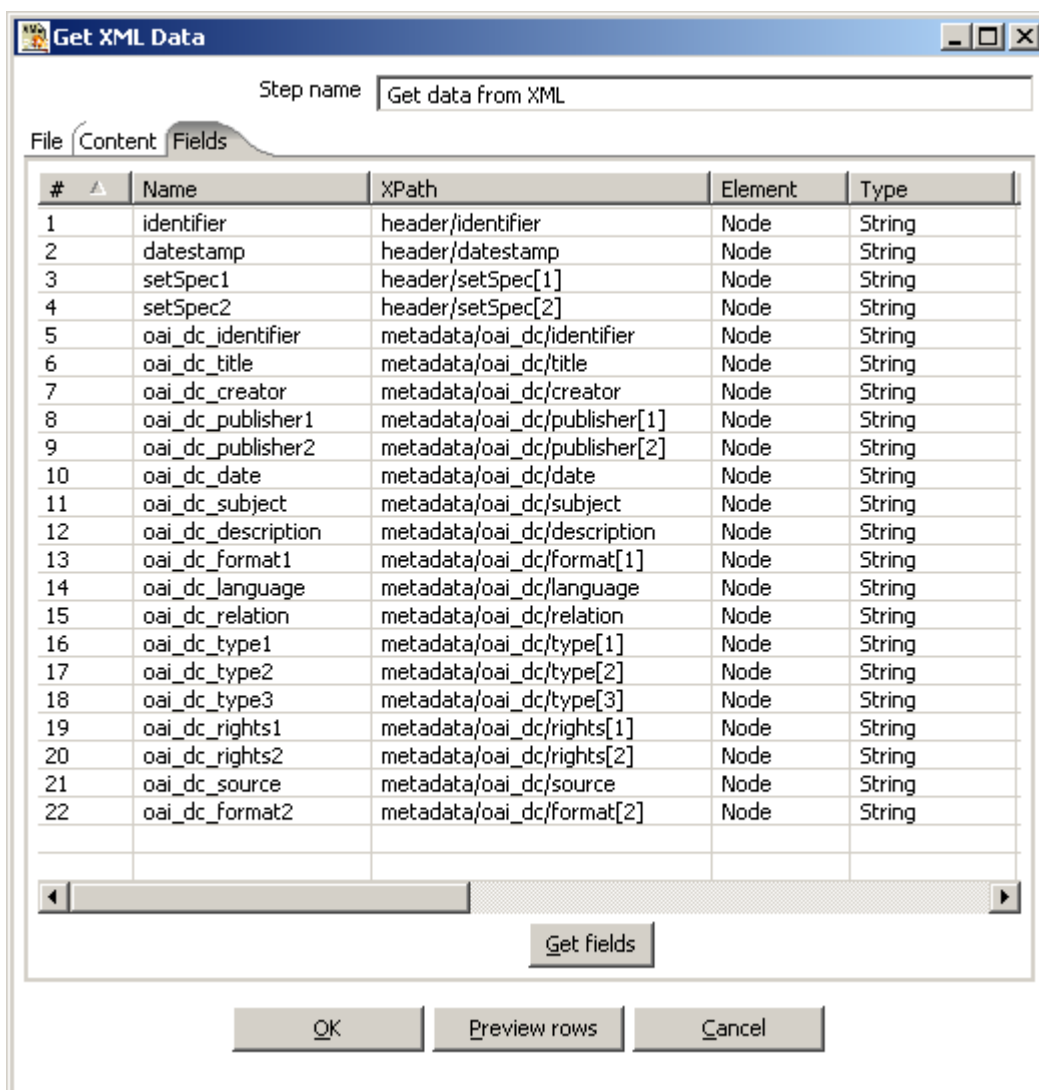


Figure 5-18 BNF transformation - Get data from XML

In the "Fields" tab the XPaths to the desired values have to be listed. To get the value of the element "identifier", "header/identifier" has to be entered, because "record" is the repeating element. Therefore the whole path does not need to be repeated again.

If there are elements with the same name, it can be selected, which to get the data from. "metadata/oai_dc/publisher[1]" gets the value of the first "publisher" element of the input XML.

The next step is a "Add constants" step. It only adds the constant value "uri", because that will be the type of the output element <mods:identifier type="uri">.

Then the basic structure of the output XML can be created using a "Add XML" step. The element names given below (Figure 5-19) are just temporary names, because to satisfy the MODS schema namespaces have to be added and for some elements child tags are required. This can not be done in this step right away, but with a small trick it can be added to the output string of this step, by replacing certain strings.

How to use this step is explained in section 3.3.1 and an example can be seen in the ISPAN transformation example (5.1).

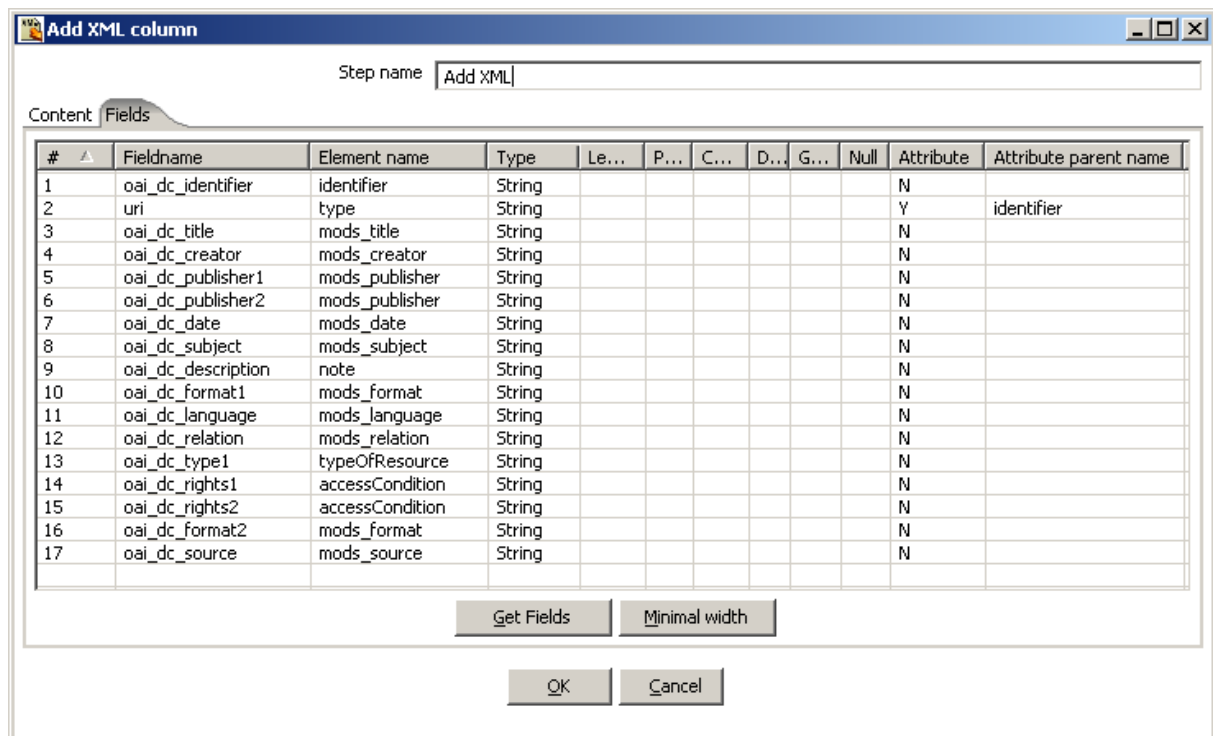


Figure 5-19 BNF transformation - Add XML step

In the "Replace string" all temporary tags are replaced by those appropriate for the MODS schema.

The outermost tag <mods> has to be replaced by this string:

```
<mods:mods version="3.3" xmlns:mods="http://www.loc.gov/mods/v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://www.loc.gov/standards/mods/v3/mods-3-3.xsd">
```

This is important that the "mods" namespace and schema are recognized.

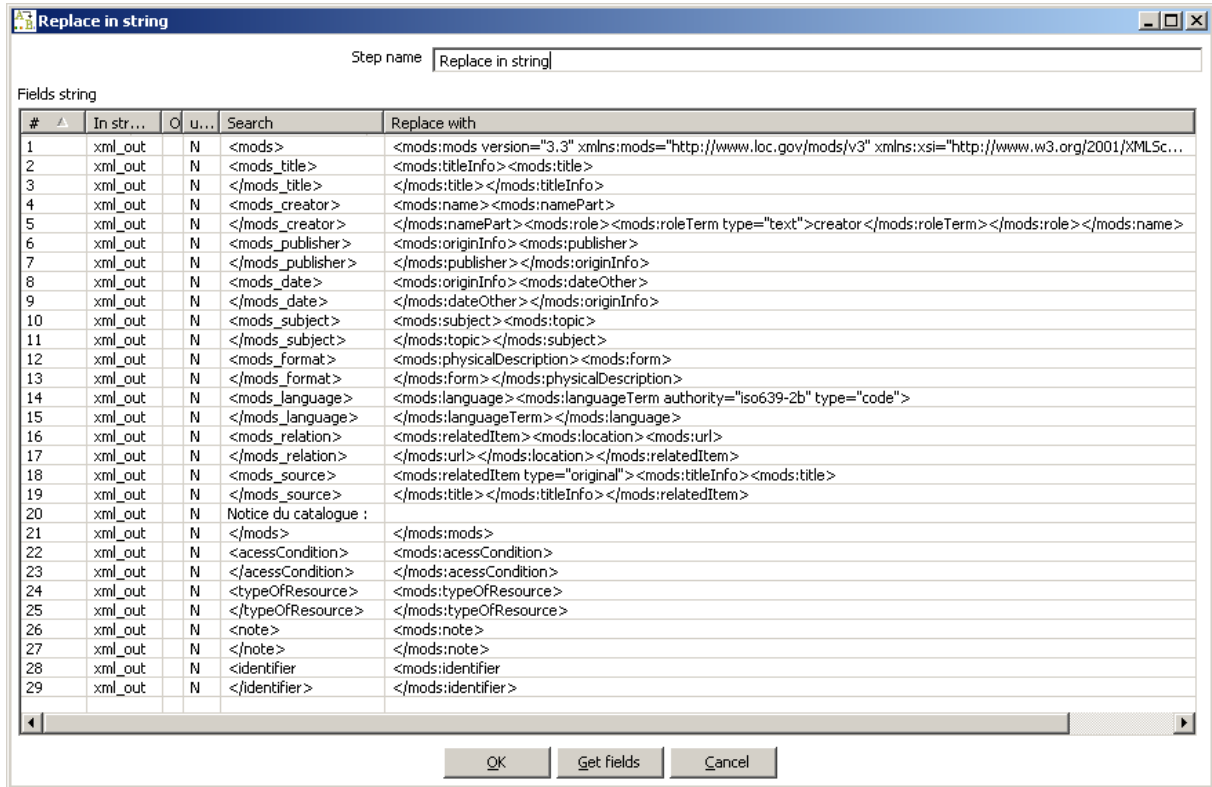


Figure 5-20 BNF transformation - replacing temporary tag with appropriate MODS tags

The final step is the text file output. Here simply the XML field from created in the "Add XML" step has to be selected. It already contains all necessary tags and information. If "Split every 1 rows" is entered a XML file will be produced for every record of the input.



Figure 5-21 BNF transformation

Figure 5-21 shows the BNF transformation with all required steps.



Glossar

ETL	<p>“Extract-Transform-Load (ETL), is a process that is used to take information from one or more sources, normalize it in some way to some convenient schema, and then insert it into some other repository.” http://www.stylusstudio.com/etl/ [Stand: 14.07.2009]</p>
PHP	<p>PHP: Hypertext Preprocessor (Personal Home Page) “PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.” http://www.php.net/ [Stand: 14.07.2009]</p>
FTP	<p>“File Transfer Protocol is a standard protocol used to exchange and manipulate files over an Internet Protocol computer network, such as the Internet.” http://en.wikipedia.org/wiki/File_Transfer_Protocol [Stand: 14.07.2009]</p>
XML	<p>“Extensible Markup Language (XML) is a simple, very flexible text format ...” “Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.” http://www.w3.org/XML/ [Stand: 14.07.2009]</p>
ISPAN	<p>Instytut Sztuki Polskiej Akademii Nauk (Institute of Art of the Polish Academy of Sciences) “The archive of the Institute of Arts of the Polish Academy of Sciences in Warsaw is the oldest and largest collection of folk music recordings in Poland.” http://www.dismarc.womex.com/html/en/archive_ispan.shtml [Stand: 14.07.2009]</p>
RegExp	<p>Regular Expression “A regular expression is a special text string for describing a search pattern.” It can for example be used to find and replace a sequence of letters in a string. http://www.regular-expressions.info/ [Stand: 14.07.2009]</p>
SSH	<p>Secure Shell is a network protocol that allows data to be exchanged using a secure channel. SSH-2 is a revised version of the protocol and is not compatible to SSH-1. http://en.wikipedia.org/wiki/Secure_Shell [Stand: 14.07.2009]</p>
SQL	<p>The Structured Query Language is a database language designed for managing data in a relational database management system. http://www.w3schools.com/SQL/sql_intro.asp [Stand 14.07.2009]</p>
MySQL	<p>MySQL is a relational database management system. http://www.w3schools.com/PHP/php_mysql_intro.asp [Stand 14.07.2009]</p>
DISMARC	<p>Discovering Music Archives http://www.dismarc.org/info/ [Stand: 14.07.2009]</p>



BNF **B**ibliothèque **n**ationale de **F**rance
National Library of France
<http://www.bnf.fr/> [Stand: 03.08.2009]

MODS **M**etadata **O**bject **D**escription **S**chema
XML based bibliographic description schema developed by the United States
Library of Congress' Network Development and Standards Office.
<http://www.loc.gov/standards/mods/>

BHL **B**iodiversity **H**eritage **L**ibrary
<http://www.biodiversitylibrary.org/> [Stand: 07.08.2009]
<http://bhl.ait.co.at/> [Stand: 07.08.2009]

CSIC **C**onsejo **S**uperior de **I**nvestigaciones **C**ientificas
Spanish National Research Council
<http://www.csic.es/> [Stand: 07.08.2009]



I. Liste verbundener Dokumente



II. List of Figures

Figure 2-1 Categories of transformation steps 6

Figure 2-2 Deleting an established hop 7

Figure 2-3 Control icons of a transformation..... 7

Figure 2-4 Control icons of a Job 7

Figure 3-1 Choosing Excel file(s) to import..... 8

Figure 3-2 Selecting sheets from an Excel file..... 9

Figure 3-3 Selecting fields from an Excel File..... 9

Figure 3-4 Preview of rows from Excel Input..... 10

Figure 3-5 Generate Rows properties..... 10

Figure 3-6 Preview of the output of the "Generate Rows" 11

Figure 3-7 Getting file names 11

Figure 3-8 Field filename with path of files 12

Figure 3-9 Filters of step "Get File Name" 12

Figure 3-10 Example XML file for "Get data from XML" step..... 13

Figure 3-11 Get data from XML properties (File) 14

Figure 3-12 Get data from XML (Available Paths) 15

Figure 3-13 - Get data from XML properties (Fields) 15

Figure 3-14 Preview of "Get date from XML" step example 15

Figure 3-15 Table input properties 16

Figure 3-16 Establishing a database connection 17

Figure 3-17 Simple transformation 17

Figure 3-18 Excel output properties..... 18

Figure 3-19 Editing content of the Excel output..... 19

Figure 3-20 Selecting fields for Excel Output 19

Figure 3-21 Text file output properties (File) 20

Figure 3-22 Text file output properties (Content) 21

Figure 3-23 XML output properties 22

Figure 3-24 Editing the content of the XML output..... 23

Figure 3-25 Selecting fields of XML output..... 23



Figure 3-26 Excel input..... 24

Figure 3-27 Sample Output of the XML output step 24

Figure 3-28 Editing the content of the "Add XML" step..... 25

Figure 3-29 Choosing fields for XML 25

Figure 3-30 Example output of the "Add XML" step (using "Text file output")..... 25

Figure 3-31 Creating constants in a Transformation..... 26

Figure 3-32 Using Calculator to merge strings 26

Figure 3-33 Predefined Calculator Operations 27

Figure 3-34 "Replace in string" properties 28

Figure 3-35 Selecting and renaming values from an input 29

Figure 3-36 Removing unnecessary Fields 29

Figure 3-37 Sorting rows by a specific field 30

Figure 3-38 Preview of input sorted by Date of Birth..... 30

Figure 3-39 Converting Date to String (meta-data) 31

Figure 3-40 Splitting the dob field into three 31

Figure 3-41 Preview of split Date of Birth field 31

Figure 3-42 Unique rows properties..... 32

Figure 3-43 XSL Transformation properties 33

Figure 3-44 Script for combining Day, Month and Year again..... 34

Figure 3-45 HTTP client properties 35

Figure 3-46 First Excel Input 35

Figure 3-47 Second Excel Input 35

Figure 3-48 Transformation to join two Excel tables 36

Figure 3-49 Configuration of the "Merge Join" step..... 36

Figure 3-50 Preview of the join of two Excel tables..... 37

Figure 3-51 Get Variables step..... 37

Figure 3-52 Set Variables..... 38

Figure 4-1 Properties of the START entry 39

Figure 4-2 Loop without DUMMY 39

Figure 4-3 Loop with DUMMY..... 40

Figure 4-4 Creating a message box notification..... 40



Figure 4-5 Message box as created in Figure 4-4 40

Figure 4-6 Example job creating a message box 41

Figure 4-7 Job calling the example job..... 41

Figure 4-8 Properties for executing a job 42

Figure 4-9 Properties for sending an e-Mail message 43

Figure 4-10 Create folder 44

Figure 4-11 Delete files..... 44

Figure 4-12 Move files 45

Figure 4-13 Simple evaluation in a job..... 46

Figure 4-14 Simple evaluation properties 46

Figure 4-15 Entering server information..... 47

Figure 4-16 Setting file options 48

Figure 4-17 File options for putting files on a FTP server 49

Figure 4-18 transformation to merge two excel sheets into one 50

Figure 4-19 Job to get files, transform them and put them back on a FTP server..... 50

Figure 4-20 Message box after successful completion of the job..... 50

Figure 4-21 Output Excel data on FTP server 51

Figure 5-1 Selected fields for dmOAP section..... 57

Figure 5-2 Required constant values..... 58

Figure 5-3 Add constant values - dmOAP attributes..... 58

Figure 5-4 constants for web-services (dmGenre and general)..... 59

Figure 5-5 Add XML (OAP) 62

Figure 5-6 XML Add (OAP) attributes..... 63

Figure 5-7 ISPAN transformation - Select for merging (OAP) 64

Figure 5-8 ISPAN transformation - Select values (oai) 64

Figure 5-9 ISPAN transformation - Add XML (oai)..... 65

Figure 5-10 ISPAN transformation - Sort rows 66

Figure 5-11 ISPAN transformation - Row flattener..... 66

Figure 5-12 ISPAN transformation - replace unwanted strings 67

Figure 5-13 ISPAN transformation - final output (xml) 68

Figure 5-14 ISPAN transformation..... 69



Figure 5-15 ISPAN Job..... 69

Figure 5-16 BNF transformation - source XML structure 70

Figure 5-17 BNF transformation - header and metadata structure 70

Figure 5-18 BNF transformation - Get data from XML 71

Figure 5-19 BNF transformation - Add XML step..... 72

Figure 5-20 BNF transformation - replacing temporary tag with appropriate MODS tags..... 73

Figure 5-21 BNF transformation..... 73

Figure 2-1 Categories of transformation steps 6

Figure 2-2 Deleting an established hop 7

Figure 2-3 Control icons of a transformation..... 7

Figure 2-4 Control icons of a Job..... 7

Figure 3-1 Choosing Excel file(s) to import..... 8

Figure 3-2 Selecting sheets from an Excel file..... 9

Figure 3-3 Selecting fields from an Excel File..... 9

Figure 3-4 Preview of rows from Excel Input..... 10

Figure 3-5 Generate Rows properties..... 10

Figure 3-6 Preview of the output of the "Generate Rows" 11

Figure 3-7 Getting file names 11

Figure 3-8 Field filename with path of files 12

Figure 3-9 Filters of step "Get File Name"..... 12

Figure 3-10 Example XML file for "Get data from XML" step..... 13

Figure 3-11 Get data from XML properties (File) 14

Figure 3-12 Get data from XML (Available Paths) 15

Figure 3-13 - Get data from XML properties (Fields) 15

Figure 3-14 Preview of "Get date from XML" step example 15

Figure 3-15 Table input properties 16

Figure 3-16 Establishing a database connection 17

Figure 3-17 Simple transformation 17

Figure 3-18 Excel output properties..... 18

Figure 3-19 Editing content of the Excel output..... 19



Figure 3-20 Selecting fields for Excel Output 19

Figure 3-21 Text file output properties (File) 20

Figure 3-22 Text file output properties (Content) 21

Figure 3-23 XML output properties 22

Figure 3-24 Editing the content of the XML output..... 23

Figure 3-25 Selecting fields of XML output..... 23

Figure 3-26 Excel input..... 24

Figure 3-27 Sample Output of the XML output step 24

Figure 3-28 Editing the content of the "Add XML" step..... 25

Figure 3-29 Choosing fields for XML 25

Figure 3-30 Example output of the "Add XML" step (using "Text file output")..... 25

Figure 3-31 Creating constants in a Transformation..... 26

Figure 3-32 Using Calculator to merge strings 26

Figure 3-33 Predefined Calculator Operations 27

Figure 3-34 "Replace in string" properties 28

Figure 3-35 Selecting and renaming values from an input 29

Figure 3-36 Removing unnecessary Fields..... 29

Figure 3-37 Sorting rows by a specific field 30

Figure 3-38 Preview of input sorted by Date of Birth..... 30

Figure 3-39 Converting Date to String (meta-data) 31

Figure 3-40 Splitting the dob field into three 31

Figure 3-41 Preview of split Date of Birth field 31

Figure 3-42 Unique rows properties..... 32

Figure 3-43 XSL Transformation properties 33

Figure 3-44 Script for combining Day, Month and Year again 34

Figure 3-45 HTTP client properties 35

Figure 3-46 First Excel Input 35

Figure 3-47 Second Excel Input 35

Figure 3-48 Transformation to join two Excel tables 36

Figure 3-49 Configuration of the "Merge Join" step..... 36

Figure 3-50 Preview of the join of two Excel tables..... 37



Figure 3-51 Get Variables step..... 37

Figure 3-52 Set Variables..... 38

Figure 4-1 Properties of the START entry 39

Figure 4-2 Loop without DUMMY 39

Figure 4-3 Loop with DUMMY..... 40

Figure 4-4 Creating a message box notification..... 40

Figure 4-5 Message box as created in Figure 4-4 40

Figure 4-6 Example job creating a message box 41

Figure 4-7 Job calling the example job..... 41

Figure 4-8 Properties for executing a job 42

Figure 4-9 Properties for sending an e-Mail message 43

Figure 4-10 Create folder 44

Figure 4-11 Delete files..... 44

Figure 4-12 Move files 45

Figure 4-13 Simple evaluation in a job..... 46

Figure 4-14 Simple evaluation properties 46

Figure 4-15 Entering server information..... 47

Figure 4-16 Setting file options..... 48

Figure 4-17 File options for putting files on a FTP server 49

Figure 4-18 transformation to merge two excel sheets into one 50

Figure 4-19 Job to get files, transform them and put them back on a FTP server..... 50

Figure 4-20 Message box after successful completion of the job..... 50

Figure 4-21 Output Excel data on FTP server 51

Figure 5-1 Selected fields for dmOAP section..... 57

Figure 5-2 Required constant values..... 58

Figure 5-3 Add constant values - dmOAP attributes..... 58

Figure 5-4 constants for web-services (dmGenre and general)..... 59

Figure 5-5 Add XML (OAP) 62

Figure 5-6 XML Add (OAP) attributes 63

Figure 5-7 ISPAN transformation - Select for merging (OAP) 64

Figure 5-8 ISPAN transformation - Select values (oai) 64



Figure 5-9 ISPAN transformation - Add XML (oai)..... 65

Figure 5-10 ISPAN transformation - Sort rows 66

Figure 5-11 ISPAN transformation - Row flattener 66

Figure 5-12 ISPAN transformation - replace unwanted strings 67

Figure 5-13 ISPAN transformation - final output (xml) 68

Figure 5-14 ISPAN transformation..... 69

Figure 5-15 ISPAN Job..... 69

Figure 5-16 BNF transformation - source XML structure 70

Figure 5-17 BNF transformation - header and metadata structure 70

Figure 5-18 BNF transformation - Get data from XML 71

Figure 5-19 BNF transformation - Add XML step..... 72

Figure 5-20 BNF transformation - replacing temporary tag with appropriate MODS tags..... 73

Figure 5-21 BNF transformation 73



III. List of Tables

Table 5-1 ISPAN mapping	55
Table 5-2 parameters for all web-services	58
Table 5-3 dmGenres parameters	59
Table 5-4 dmFormats parameters	59
Table 5-5 dmGeography parameters	60
Table 5-6 dmEras parameters.....	60