

# DELIVERABLE REPORT

**Project Acronym:** OpenUp!

**Grant Agreement No:** 270890

**Project Title:** Opening up the Natural History Heritage for Europeana

## D02/D2.2.2 Harvesting and Transformation Component production version

### Part 3: The Pentaho Transformation Procedure

**Revision:** Doc02a - Final

**Authors:**

Astrid Höller                    AIT Forschungsgesellschaft mbH

Odo Benda                        AIT Forschungsgesellschaft mbH

Walter Koch                      AIT Forschungsgesellschaft mbH

.....

<b>Project co-funded by the European Commission within the ICT Policy Support Programme</b>		
<b>Dissemination Level</b>		
<b>P</b>	Public	<b>x</b>
<b>C</b>	Confidential, only for members of the consortium and the Commission Services	

## Revision History

Revision	Date	Author	Organisation	Description
01	13.5.2011	Walter Koch	AIT	The OpenUp! harvesting prototype component installed with the help of: Odo Benda (AIT), Gerda Koch (AIT), Bernd Sproger (AIT)
01a	20.5.2011	Walter Koch	AIT	Start of Harvesting tests with BGBM data
01b	8.6.2011	Walter Koch	AIT	Start of Harvesting tests with NHMW data
01c	30.6.2011	Walter Koch	AIT	Start of Harvesting tests with MfN data
02	2.8.2011	Walter Koch	AIT	Virtual machine with harvesting prototype for test established and provided for the Work package leader (test111.ait.co.at with GBIF-Harvester and metadata manager)
02a	19.8.2011	Walter Koch	AIT	Start of Harvesting tests with BGBM Herbar data (test111.ait.co.at:8080/hit). Start of tests for Europeana thumbnail generation
02b	24.9.2011		UT-NHM	Harvesting test by UT-NHM
03	20.11.2011	Walter Koch	AIT	Virtual Machine with: GBIF-Harvester ( <a href="http://test117.ait.co.at:8080/hit/">http://test117.ait.co.at:8080/hit/</a> ) OAI-Provider <a href="http://test117.ait.co.at/oai-provider/index.php?form=index&amp;db=0">http://test117.ait.co.at/oai-provider/index.php?form=index&amp;db=0</a> Pentaho with jobs and transformation is also installed. On this machine we harvested appr. 200.000 ABCD-units which have been transformed (1st draft) to ESE 3.4 (can be searched and inspected via the OAI-Provider)
Doc01	17.11.2011	Walter Koch, Astrid Höller, Odo Benda, Gerda Koch	AIT	Draft of documentation (Part 1 to 3)
Doc02	18.11.2011	Gerda Koch	AIT	Version 1 of documentation (Part 1 to 3), formatting for distribution
Doc02a	21.11.2011	Coordination Team	BGBM	Minor editing

### Statement of Originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

## Distribution

Recipient	Date	Version	Accepted YES/NO
TMG	Continuous	WebSite: <a href="http://open-up.eu/content/harvesting-and-transformation-component">http://open-up.eu/content/harvesting-and-transformation-component</a>	
Work Package Leader	2.8.2011	Virtual machine with harvesting prototype	
BGBM	19.8.2011	Feedback and communication on Harvesting Tests with HERBAR	
Work Package Leader	20.11.2011	Virtual Machine with: GBIF-Harvester ( <a href="http://test117.ait.co.at:8080/hit/">http://test117.ait.co.at:8080/hit/</a> ) OAI-Provider <a href="http://test117.ait.co.at/oai-provider/index.php?form=index&amp;db=0">http://test117.ait.co.at/oai-provider/index.php?form=index&amp;db=0</a> Pentaho with jobs and transformation is also installed.  On this machine we harvested appr. 200.000 ABCD-units which have been transformed (1st draft) to ESE 3.4 (can be searched and inspected via the OAI-Provider)	
Work Package Leader	18.11.2011	Doc02 (Part 1 to 3)	WP leader G. Malarky and the Technology Management Group have been constantly following the progress of this Deliverable. Unfortunately, Mr. Malarky is not available until the end of November for providing his final acceptance.
Project Coordinator	18.11.2011	Doc02a (Part 1 to 3)	Yes

## Table of Contents

<b>1. Management Summary .....</b>	<b>6</b>
1.1. Access to Biological Collection Data (ABCD) .....	6
1.2. Europeana v1.0 project .....	6
1.2.1 The ESE v3.4 XML Schema .....	7
1.3. Pentaho Data Integration.....	7
<b>2. Pentaho Kettle Data Integration – User Interface .....</b>	<b>9</b>
<b>3. From ABCD to ESE with Pentaho .....</b>	<b>11</b>
J.1 The Pentaho Job “Biocase_Harvest_to_ESE” (I) .....	11
T.1 Get_Files_from_HIT_DB (AIT111) .....	12
T.1.1 Table Input - From HIT DB AIT111 .....	13
T.1.2 Json Input – get parameter, get directory .....	15
T.1.3 Select values – drop json .....	16
T.1.4 Add constants – set file pattern search_response *.gz.....	17
T.1.5 Replace in String – build filesystem path .....	18
T.1.6 Get File Names .....	19
T.1.7 Select Values – Select result .....	21
T.1.8 Copy Rows to Result.....	22
T.2 adapt_ABCD_load_for_ese_transformation.....	23
T.2.1 Get rows from result.....	25
T.2.2 Replace in string .....	25
T.2.3 Select values.....	26
T.3 The “ABCD206_to_ESE3” Transformation.....	27
T.3.1 Get rows from result – Input filenames .....	28
T.3.2 Generate rows – Test data .....	29
T.3.3 Load file content in memory – Read ABCD xml file .....	30
T.3.4 Get XML data – Get Units from XML .....	32
T.3.5 Add constants – ABCD to ESE XSL.....	34
T.3.6 XSL Transformation.....	35
T.3.7 Select values – remove input file.....	37

T.3.8	Add constants – Add result field .....	38
T.3.9	Filter Rows .....	39
T.3.10	Modified Java Script Value – err-filename from UnitId.....	40
T.3.11	Modified Java Script value – filename from UnitID.....	41
T.3.12	Text file output – Save erroneous ESE records .....	42
T.3.13	Text file output – save ESE records.....	44
T.3.14	Set field value to constant – set OK .....	46
T.3.15	Add constant value –set reason OK.....	47
T.3.16	Add constants – Add result field 2.....	48
T.3.17	“Select values” – remove input file 2.....	49
T.3.18	“set field value to a constant” – set NOT OK.....	50
T.3.19	Get data from XML – Get error Reason .....	52
T.3.20	Select values –select result.....	54
T.3.21	Modified Java Script value – logfilename with date .....	55
T.3.22	Text file output – Log: ABCD206_to_ESE34_<date>.csv .....	56
T.3.23	Copy rows to result .....	59
J.1	The Pentaho Job “Biocase_Harvest_to_ESE” (II) .....	59
<i>J.1.T.1</i>	<i>Get_Files_from_HIT_DB (AIT111) and START .....</i>	<i>60</i>
J.1.1	Write to file – create extract-abcd.sh .....	62
J.1.2	Shell – chmod + x extract-abcd.sh .....	63
J.1.3	Shell – extract-abcd.sh .....	65
<i>J.1.T.2</i>	<i>Transformation – adapt_abcd_load_for_ese_transform.....</i>	<i>67</i>
<i>J.1.T.3</i>	<i>Transformation – ABCD206_to_ESE34 .....</i>	<i>68</i>
<b>4.</b>	<b>Pentaho example .....</b>	<b>71</b>
4.1.	The ABCD example record .....	71
4.2.	The results of J.1 “Biocase_Harvest_to_ESE” .....	72
4.2.1	<i>Created folder structure.....</i>	<i>73</i>
4.2.2	<i>The .tmp directory .....</i>	<i>75</i>
<b>Glossar</b> .....		<b>77</b>
<b>I. List of Figures</b> .....		<b>79</b>
<b>II. List of Pentaho Figures</b> .....		<b>80</b>

## 1. Management Summary

This document illustrates the use of Pentaho Kettle Data Integration (DI) with examples of the OpenUp! project. Similarly to the PHP mappers, Kettle is used to import data of different formats (Excel sheets, databases, XML documents), transform it and export it again via FTP. A summary of Kettle DI is given and various transformation steps are explained. The main transformation of this document is used to transform ABCD ('Access to Biological Collection Data) documents to XML files which follow the Europeana standard. Further information on these standards will be given in the next sections.

### 1.1. *Access to Biological Collection Data (ABCD)*

The Access to Biological Collections Data (ABCD)<sup>1</sup> Schema is an evolving comprehensive standard for the access to and exchange of data about specimens and observations (a.k.a. primary biodiversity data).

The ABCD Schema<sup>2</sup> attempts to be comprehensive and highly structured, supporting data from a wide variety of databases. It is compatible with several existing data standards. Parallel structures exist so that either (or both) atomized data and free-text can be accommodated. Version 2.06 is currently in use with the GBIF (Global Biodiversity Information Facility<sup>3</sup>) and BioCASE (Biological Collection Access Service for Europe<sup>4</sup>) networks.

### 1.2. *Europeana v1.0 project*

The objectives of the Europeana Foundation are:

- providing access to Europe's cultural and scientific heritage by way of a cross-domain portal
- facilitating formal agreement across museums, archives, audiovisual archives and libraries on how to co-operate in the delivery and sustainability of a joint portal
- stimulating and facilitating initiatives to bring together existing digital content
- supporting and facilitating digitization of Europe's cultural and scientific heritage<sup>5</sup>

---

<sup>1</sup> <http://wiki.tdwg.org/twiki/bin/view/ABCD/AbcdPrimer>

<sup>2</sup> <http://www.tdwg.org/standards/115/>

<sup>3</sup> <http://www.gbif.org/>

<sup>4</sup> <http://www.biocase.org/>

<sup>5</sup> <http://version1.europeana.eu/web/europeana-project/home> )

### 1.2.1 The ESE v3.4 XML Schema

The ESE v3.4 XML Schema<sup>6</sup> is the XML representation of the Europeana Semantic Elements (ESE) specifications v3.4. This schema can be used to validate XML instances of Data Sets to be submitted to Europeana. The ESE v3.4 XML Schema extends the DC XML Schema<sup>7</sup> with the addition of elements belonging to the Europeana namespace.

A typical XML instance file containing ESE metadata records has the following structure:

```
<metadata xmlns="http://www.europeana.eu/schemas/ese/"
xmlns:europeana="http://www.europeana.eu/schemas/ese/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms/">
<record>
<europeana:isShownBy>http://www.xx.yy/yy</europeana:isShownBy>
<dc:subject>archéologie ; Grec ; Céramique</dc:subject>
...
</record>
<record>
...
</record>
...
</metadata>
```

Figure 1–1 The ESE Schema

### 1.3. Pentaho Data Integration

Pentaho Data Integration is an ETL (Extract Transform Load) tool with a graphical user interface (see Figure 1–2), 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 a **Transformation** various Transformation steps can be applied to imported data (e.g. tables from an Excel sheet or XML documents). 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.

<sup>6</sup> <http://www.europeana.eu/schemas/ese/ESE-V3.4.xsd>

<sup>7</sup> <http://dublincore.org/>

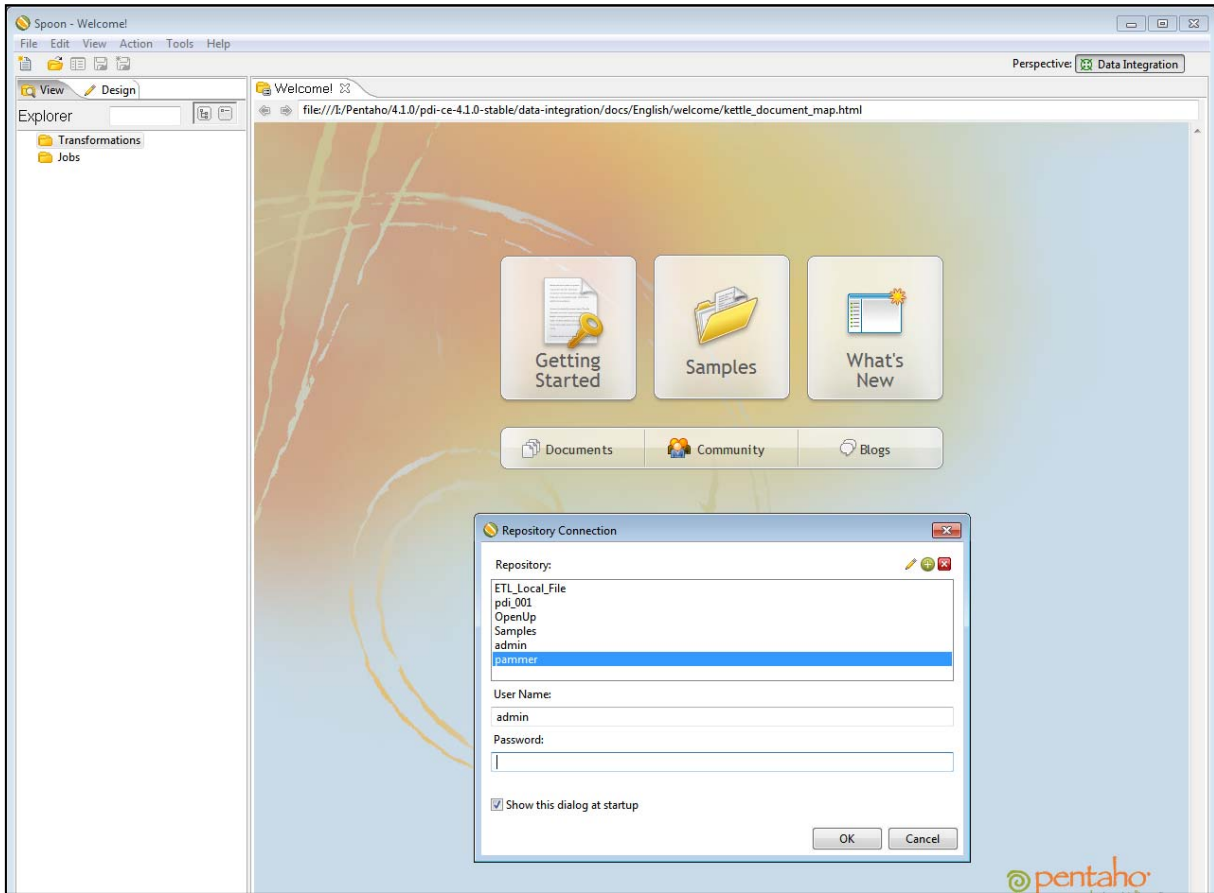


Figure 1–2 Pentaho Welcome Screen with Repository Connection window

Before you can start building Transformations or Jobs you have to log in to have a Repository Connection. In the Repository Connection window (see Figure 1–3) you can choose a Repository from the list, edit (pencil symbol) or delete (red cube) it, or create a new one (green plus).



Figure 1–3 Repository Connection window



## 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 (see Figure 2–1).

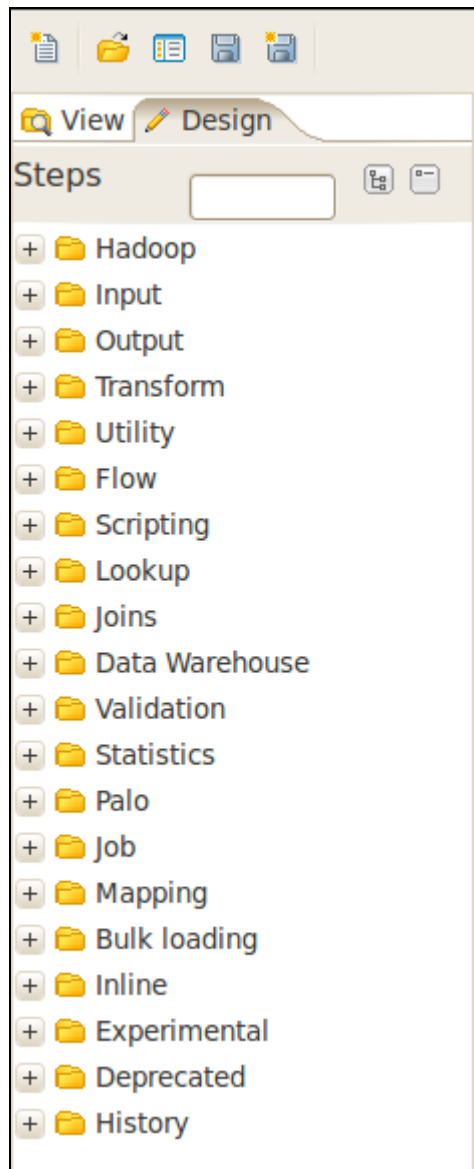


Figure 2–1 Categories of Transformation steps

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 steps. 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 to change the orientation of the connection if needed (see Figure 2–2).

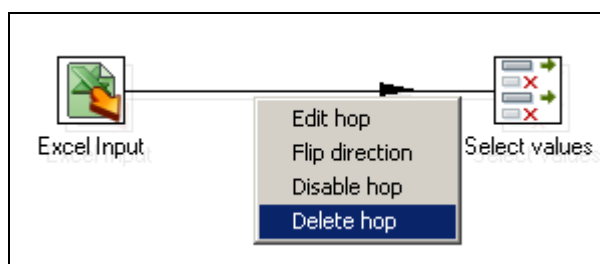


Figure 2–2 Deleting an established hop

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 (see Figure 2–3).



Figure 2–3 Control icons of a transformation

In a job there is also an icon to run and stop the current Job (see Figure 2–4).



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. From ABCD to ESE with Pentaho

This chapter will describe the Job and Transformations that will be necessary to create correct ESE data. With a Job in Pentaho you can use, modify and combine created Transformations. It is the main process which manages the Transformations.

In this documentation a Job has the notation J with following numbers, a Transformation T. The notation will be constantly used in every Figure for better understanding.

#### J.1 The Pentaho Job "Biocase\_Harvest\_to\_ESE" (I)

In Figure J.1 the Job to transform ABCD documents into ESE files can be seen. Like every Job there is a "START" and "Success" icon. Furthermore the three Transformations "Get\_Files\_from\_HIT\_DB (AIT111)", "adapt\_abcd\_load\_for\_ese\_transformation" and "ABCD206\_to\_ESE34" are included. Together with the additional steps in this Job they will be described in the next section. The colored squares are short descriptions for each process.

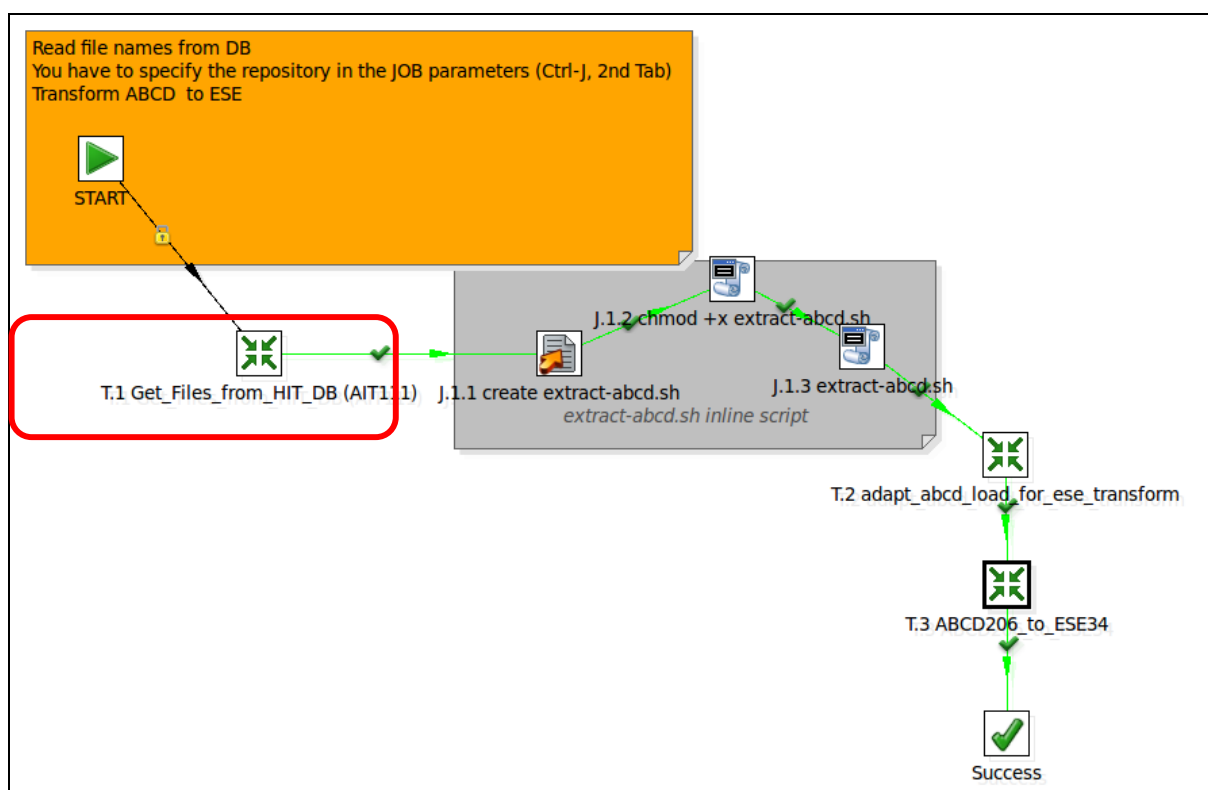
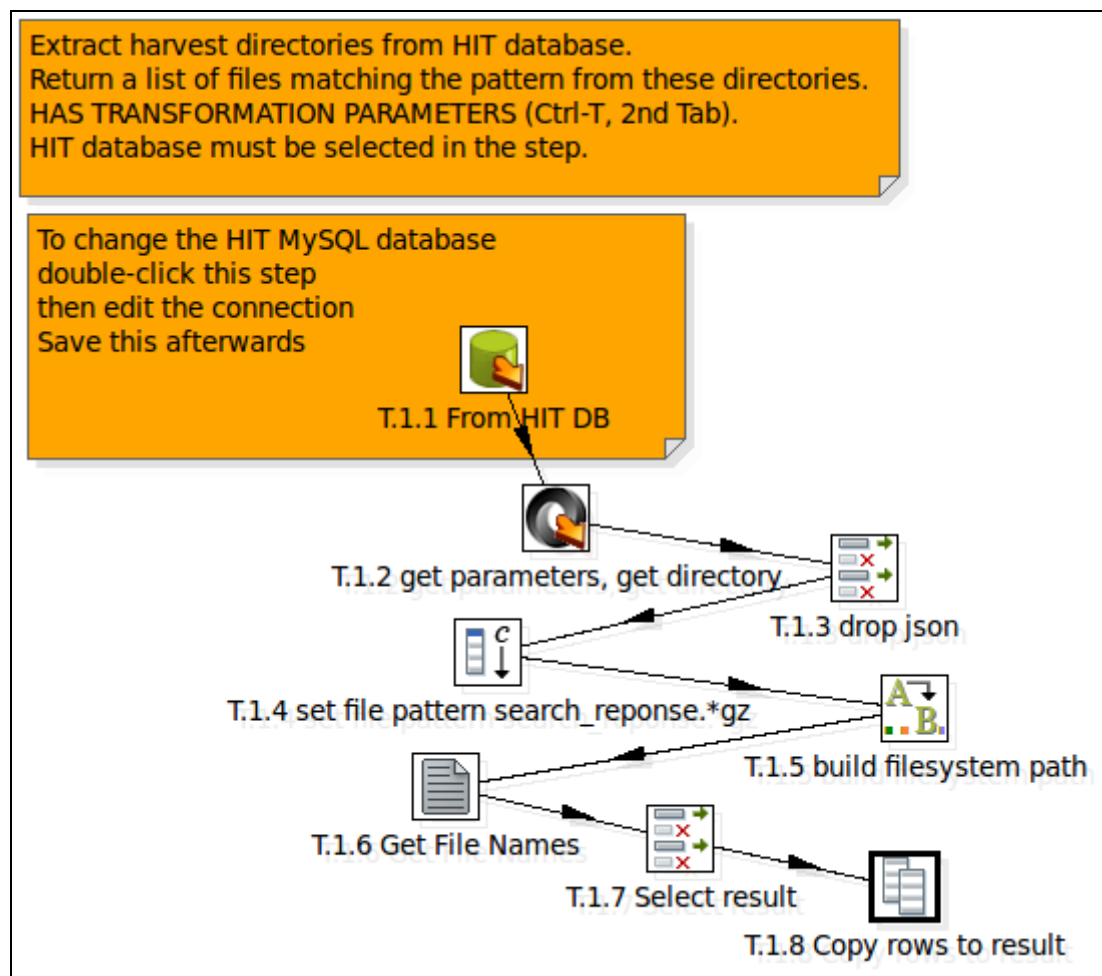


Figure J. 1 Job in Pentaho including three transformations

### T.1 *Get\_Files\_from\_HIT\_DB (AIT111)*

This Transformation is the first one in the Pentaho Job (marked red in *Figure J.1*). Before it can be included in the Job, it has to be created separately. *Figure T.1* gives an overview of the whole Transformation.



*Figure T. 1 Get\_Files\_from\_HIT\_DB Transformation*

The aim of this Transformation is to extract harvest directories from the HIT database and create an output in form of all files that match the pattern from these directories (compare *Figure T.1*). In the following sections every step will be described separately.

### T.1.1 Table Input<sup>8</sup> - From HIT DB AIT111

The first step is “Table Input”, found under the “Input” directory. In our example the step was renamed to “From HIT DB AIT111” to make clear where the data is coming from. Just drag the icon into the canvas. (Figure T.1.1).



Figure T.1.1 Table Input

When you double click on the icon the menu of this step opens (Figure T.1.1a).

Step name

Connection  Edit... New...

SQL Get SQL select statement...

```
SELECT
  parameters_as_json
FROM
  bio_datasource
WHERE
  uddi_key LIKE '${uddi_key}' AND name LIKE '${name}'
```

Line 1 Column 0

Enable lazy conversion

Replace variables in script

Insert data from step

Execute for each row?

Limit size

OK Preview Cancel

Figure T.1.1. a Define Table Input

<sup>8</sup> <http://wiki.pentaho.com/display/EAI/Table+Input>

To get the date first of all you have to select a database connection. Either you can create a new database by selecting "New" or "Edit" a database connection that already exists. In our case the database connection is defined like in Figure T.1.1b.

When the database connection has been selected the SQL Statement has to be filled like in Figure T.1.1a. It defines which data has to be selected from the table. In our example we select the field "parameters\_as\_json" from the table "bio\_datasource" where the fields "uddi\_key" and "name" meet certain conditions. The operator LIKE searches for a specific pattern in a column<sup>9</sup>. The dollar sign (\$) represents zero or one character. Figure T.1.1c gives you an example of these fields.

Do not forget to activate the option "Replace Variables in Script". When you have done that, click on "OK" and the first (input) step of the Transformation is finished.

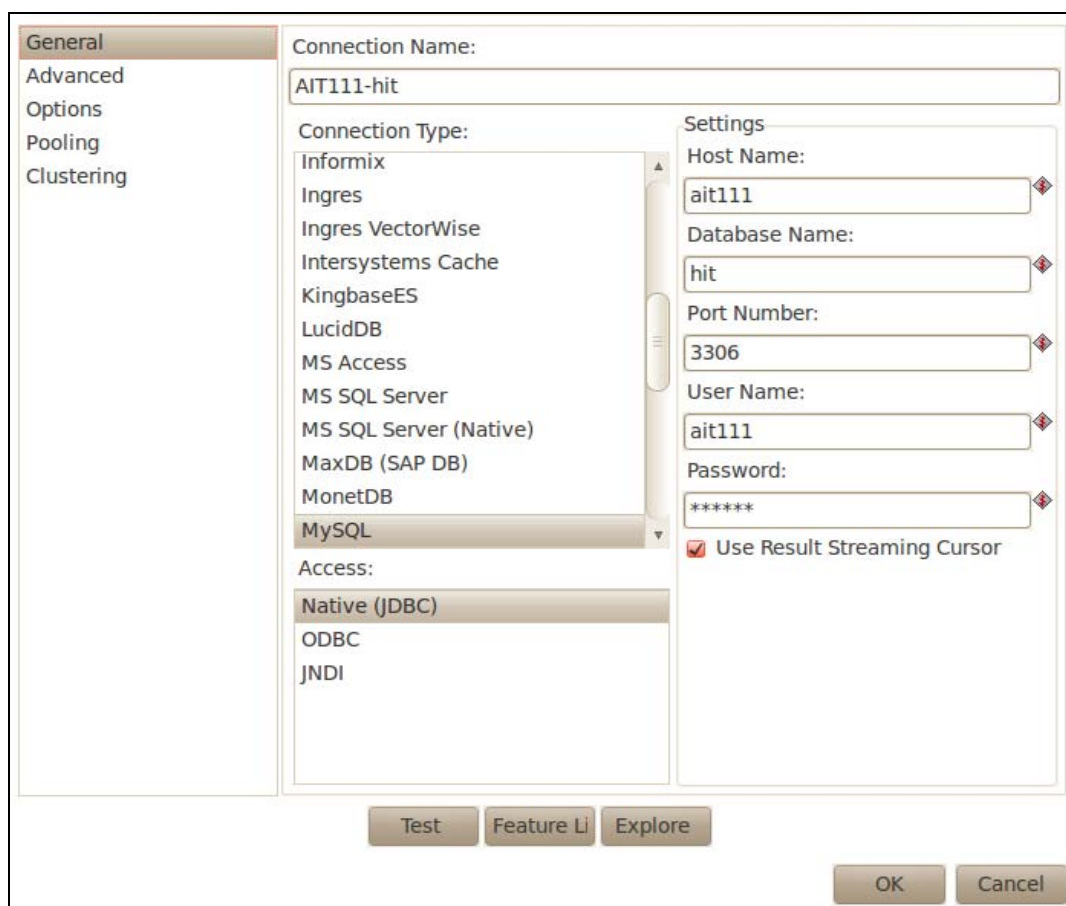


Figure T.1.1. b Database connection

<sup>9</sup> [http://www.w3schools.com/sql/sql\\_like.asp](http://www.w3schools.com/sql/sql_like.asp)

uddi_key	name	parameters_as_json
c1132a05-bcde-4fa5-a0ee-1b77ec16adc4	c1132a05	{"dataResourceName":"\"null\"","dataResourceWebsit...

Figure T.1.1. c Fields “uddi\_key”, “name” and “parameters\_as\_json” from the HIT DB

### T.1.2 Json Input – get parameter, get directory

The next step is called “Json Input” and has to be connected to the first step via hop (Figure T.1.2). It extracts relevant portions out of JSON<sup>10</sup> structures (file or incoming field) and output rows.

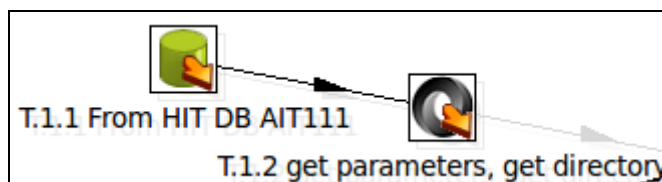


Figure T.1.2 “Json Input” connected to “Table Input”

Like with every other step you can open the menu by double-clicking on the icon (Figure T.1.2a). There you can rename the step. In our example it is called “get parameters, get directory”.

#	File/Directory	Wildcard (RegExp)	Exclude wildcard	Required	Include subfolders
1				N	N

Figure T.1.2 a File section – Json Input

As you can see in Figure T.1.2a there are four different sections: File, Content, Fields and Additional output fields. In our example we need the first three. In the File section you have to select “Source is defined in a field” and chose our Table Input field “parameters\_as\_json” after “Get source from field” because this field contains the Json data.

<sup>10</sup> <http://en.wikipedia.org/wiki/JSON>

After that you can continue with the Content section (Figure T.1.2b). There you can define the Settings. Make sure the “Ignore missing path” option is activated.

Figure T.1.2.b Content section – Json Input

In the Fields section you can finally choose the fields you get from your database (Figure T.1.2c). You have to type them by yourself. Do not forget to define the type for each field.

#	Name	Path	Type	Format	Length	Precision	Currency	Decimal	Group
1	directory	directory	String						
2	providerUuid	providerUuid	String						
3	targetCount	targetCount	Integer						
4	startLastSync	startLastSync	Integer						
5	contentNamespace	contentNamespace	String						
6	country	country	String						
7	countryCode	countryCode	String						
8	resource_name	resource_name	String						
9	uddiKey	uddiKey	String						

Figure T.1.2c Field section – Json Input

### T.1.3 Select values<sup>11</sup> – drop json

The Select values step (renamed “drop json”) is the third step in our Transformation. It has to be connected with the previous step (Figure T.1.3)

<sup>11</sup> <http://wiki.pentaho.com/display/EAI/Select+Values>



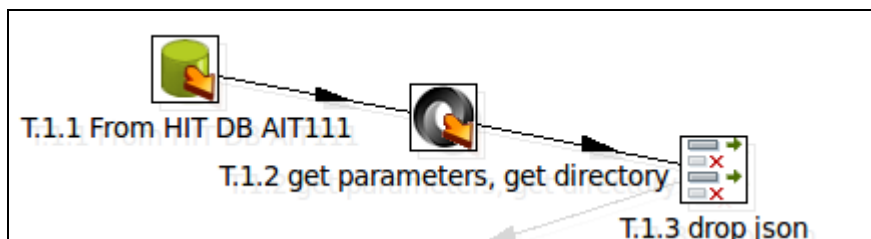


Figure T.1.3 Connection to the third step Select values – “drop json”

With this step it is possible to rename, select or delete the fields from the previous step. We use it to remove the field “parameters\_as\_json” because after getting the fields we will use later we do not need it anymore. Note that there are three different sections in the Select value step. Make sure you choose the second one “Remove” in this step (see Figure T.1.3a).

You can either click on “Get fields to remove” and delete the other fields or you use the drop down menu in the field and choose “parameters\_as\_json”.

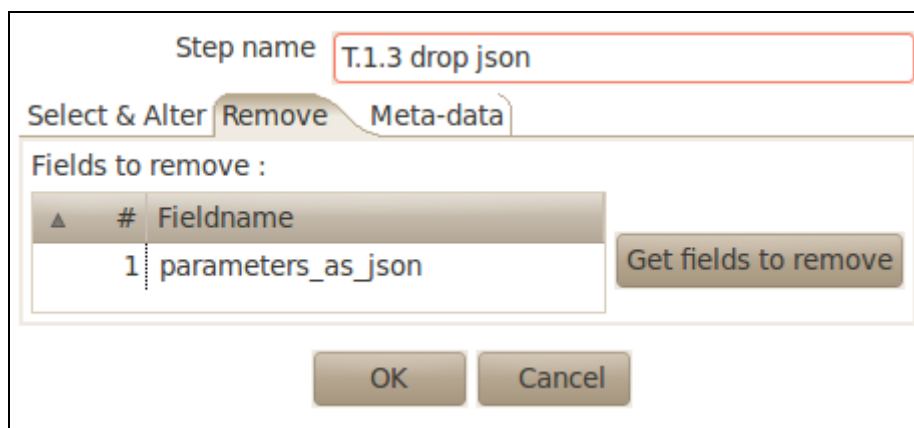


Figure T.1.3a Select Values – Remove “parameters\_as\_json”

#### T.1.4 Add constants<sup>12</sup> – set file pattern search\_response \*.gz

Again the new step “Add constants” has to be connected with the last one (Select Values) via hop (Figure T.1.4).

<sup>12</sup> <http://wiki.pentaho.com/display/EAI/Add+Constants>

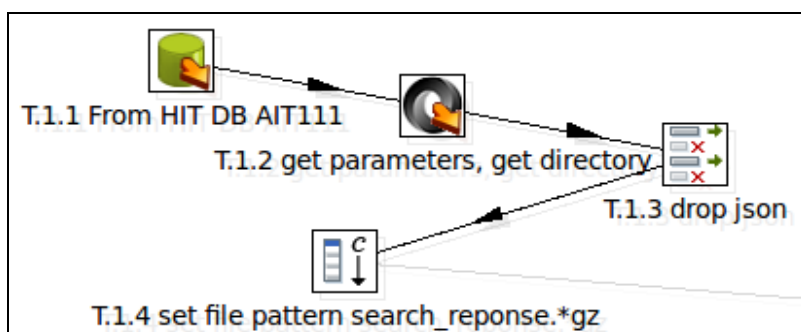


Figure T. 1. 4 Adding the “Add Constants” step

When double-clicking on the icon “Add constants” you can see a fields sections. (Figure T. 1. 4a):

Step name

Fields :

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value
1	pattern	String							search_response.*gz
2	path	String							:base_dir:/directory

OK Cancel

Figure T. 1. 4a Add the constants “pattern” and “path”

With this step you can add values that stay constant during the Transformation. In our example it is used to determine the pattern and the path of the files which they have later to match. So you have two fields: “pattern” and “path”. The values are “*search\_response.\*gz*” and “*:base\_dir:/directory*”. They have to be entered under “Value”. Now we know that our files will have the name “*search\_response.\*gz*”. The \* sign is used as wildcard expression for different numbers. “gz” is the filename extension for GNU zip – a file type for data compression.<sup>13</sup>

When you have entered the values, click “OK” and the step is finished.

In order to create the correct paths we need another step: Replace in String.

### T.1.5 **Replace in String<sup>14</sup> – build filesystem path**

First of all you have to connect this step with the previous one (Figure T. 1. 5).

<sup>13</sup> <http://en.wikipedia.org/wiki/Gzip>

<sup>14</sup> <http://wiki.pentaho.com/display/EAI/Replace+in+String>

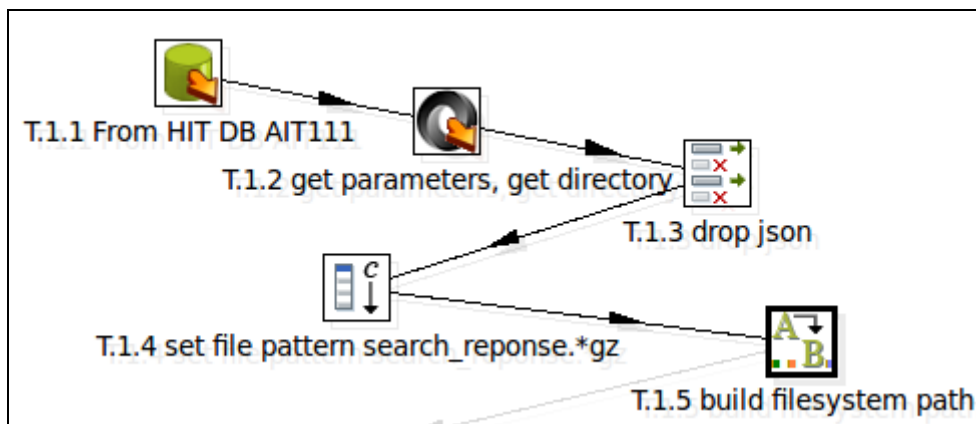


Figure T.1.5 Adding the “Replace in String” step

“Replace in String” is a simple search and replace function. In Figure 3–16 you can see the settings of the Replace in String step.

Step name: T.1.5 build filesystem path

Fields string

#	In stream field	Out stream field	use RegEx	Search	Replace with	Replace with field	Whole Word	Case sensitive
1	path		N	:base_dir	\${base_dir}		N	Y
2	path		N	:directory		directory	N	Y

OK Get fields Cancel

Figure T.1.5a Replace in String

In this step the file path defined in the last step is splitted into two parts and is replaced with parameters in order to create files in the correct directory. In Figure T.1.5a you can see that `:base_dir` is replaced with the parameter `${base_dir}` and `:directory` is replaced with the values of the field `directory`. When you have filled in everything correctly click “OK”.

### T.1.6 Get File Names<sup>15</sup>

This step again is connected with the last one via hop. At this point the Transformation should look like the one in Figure T.1.6.

<sup>15</sup> <http://wiki.pentaho.com/display/EAI/Get+File+Names>

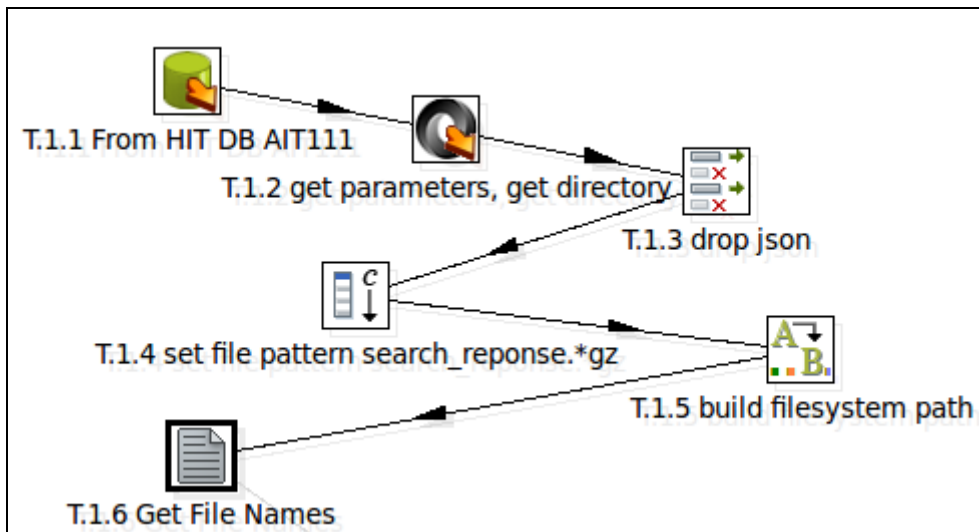


Figure T.1.6 Adding the "Get File Names" step

This step gets file names from the operating system and sends them to the next step. In order to achieve this, your settings should comply with the one in Figure T.1.6a and Figure T.1.6b.

Figure T.1.6a File section of the step Get File Names

In the File section (Figure T.1.6a) of this step you have to activate the option "Filename is defined in a field" and choose the "path" field in the next line ("Get filename from field") because there is the data we get the filename from. Another important field is "pattern", which has to be selected for "Get wildcard from field (RegExp)". Remember that our pattern was "search\_response.\*gz" with the \* sign representing the wildcard expression.

Figure T.1.6b Filters section of the step Get File Names

In the Filters section (see Figure T.1.6b) you have to make sure that you select “All Files” after “Get” because we want to work with all files and “Add filename to result” is NOT activated.

When this is done you can click on “OK” and add another Select Values step.

### T.1.7 Select Values – Select result

After adding this step, the Transformation should look like in Figure T.1.7. We need this step again to determine which fields we finally want to work with in our Pentaho Job.

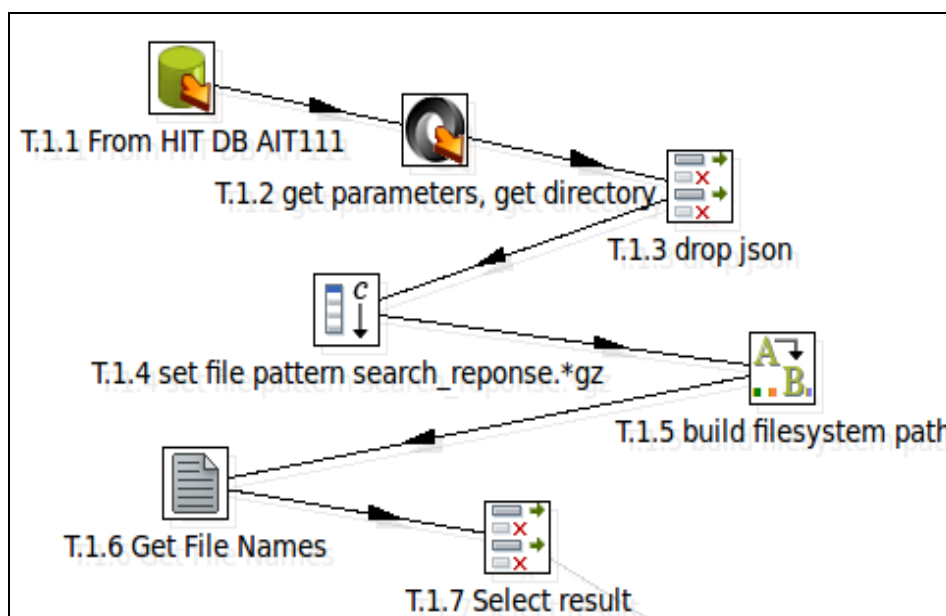
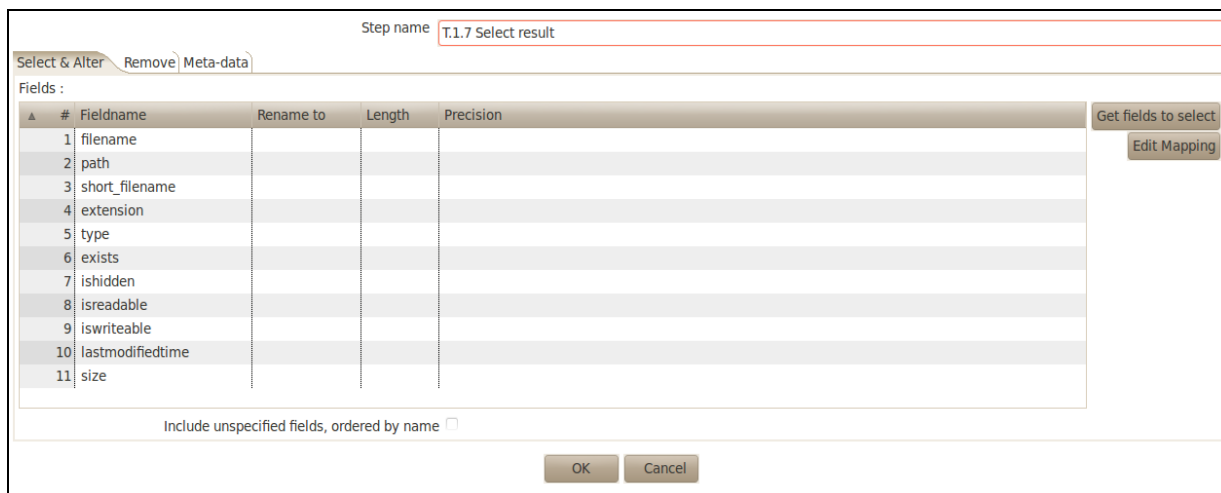


Figure T.1.7 Adding another “Select Values” step

When you click on “Get fields to select” in the menu of this step, all fields from the previous step are chosen. *Figure T.1.7a* shows the list of fields you need to have after this step. Simply delete the ones not needed. Now only one step is missing to complete the first transformation.



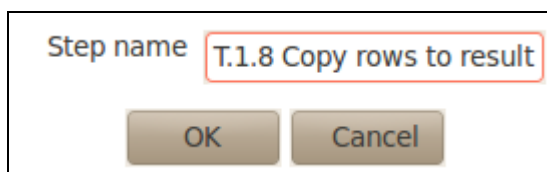
*Figure T.1.7a Get fields to select*

### T.1.8 Copy Rows to Result<sup>16</sup>

When this step is added the transformation should look like in T.1.

This step is used to write rows to the executing job. The information will then be passed to the next entry in this job. So all fields we have chosen before will be used in the Job.

There are no settings for this step. When you double-click on the icon, the following window will appear (*Figure T.1.8a*). If you want you can rename the step.



*Figure T.1.8 Copy Rows to Result*

For most Transformations preview data is available. If you click on the preview symbol while step T.1.8 is selected the following window appears (see *Figure T.1.8a*). It shows the filenames and their paths plus short\_filename, extension etc.

<sup>16</sup> <http://wiki.pentaho.com/display/EAI/Copy+rows+to+result>

Rows of step: T.1.8 Copy rows to result (1000 rows)

#	filename	path	short_filename	exte	type
1	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.000.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.000.gz	gz	file
2	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.001.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.001.gz	gz	file
3	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.002.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.002.gz	gz	file
4	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.003.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.003.gz	gz	file
5	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.004.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.004.gz	gz	file
6	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.005.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.005.gz	gz	file
7	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.006.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.006.gz	gz	file
8	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.007.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.007.gz	gz	file
9	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.008.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.008.gz	gz	file
10	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.009.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.009.gz	gz	file
11	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.010.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.010.gz	gz	file
12	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.011.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.011.gz	gz	file
13	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.012.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.012.gz	gz	file
14	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.013.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.013.gz	gz	file
15	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.014.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.014.gz	gz	file
16	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.015.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.015.gz	gz	file
17	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.016.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.016.gz	gz	file
18	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.017.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.017.gz	gz	file
19	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.018.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.018.gz	gz	file
20	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.019.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.019.gz	gz	file
21	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.020.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.020.gz	gz	file
22	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.021.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.021.gz	gz	file
23	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.022.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.022.gz	gz	file
24	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense/search_response.023.gz	/opt/hit/4161387-bgbm-herbar/herbarium_berolinense	search_response.023.gz	gz	file

Close Stop Get more rows

Figure T. 1.8a Rows to display during preview

## T.2 adapt\_ABCD\_load\_for\_ese\_transformation

After completing the first Transformation we need for the Job we continue with the second one. In our example it is called "adapt\_ABCD\_load\_for\_ese". Figure J.2 shows the job again with this Transformation we are creating now marked red.

The aim of this Transformation is to adapt the filenames and output directories. Figure T.2 gives an overview of the whole Transformation.

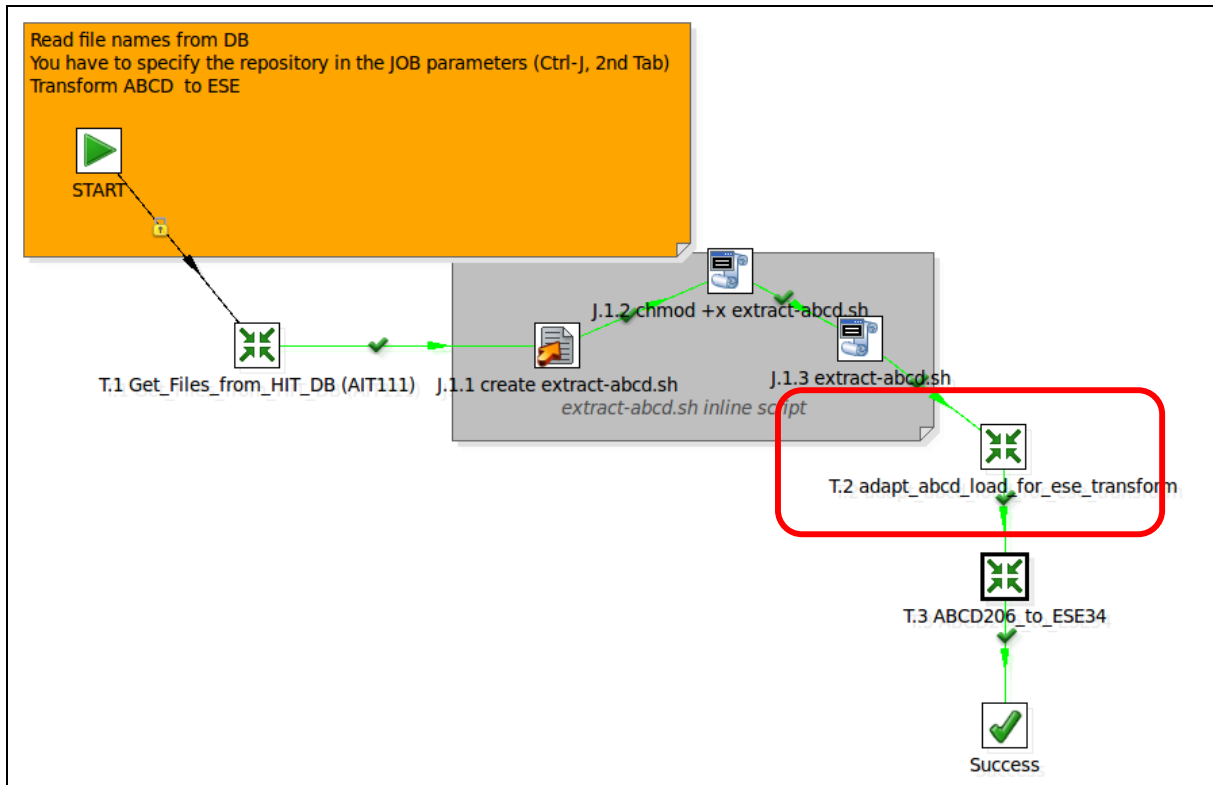


Figure J.2 The Pentaho Job

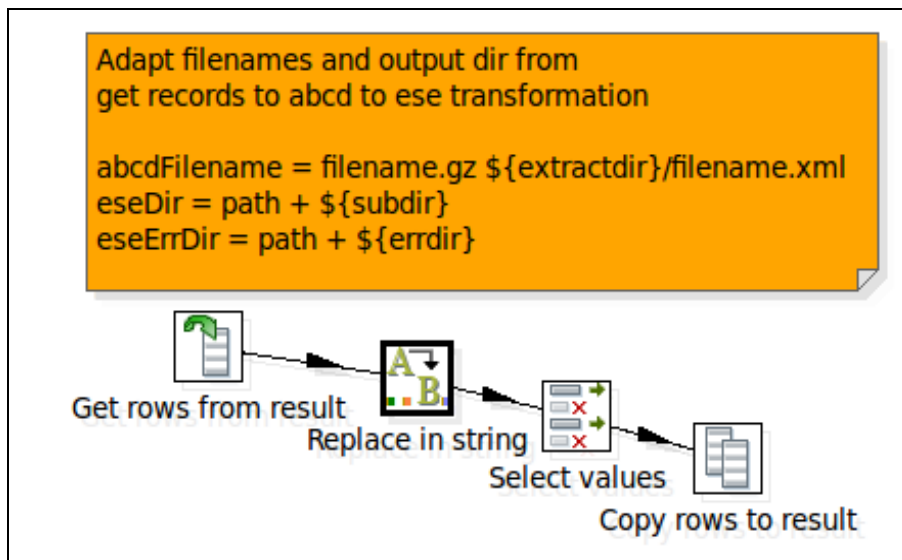


Figure T.2 The "adapt\_ABCD\_load\_for\_ese\_transform" Transformation



### T.2.1 Get rows from result<sup>17</sup>

This step returns rows that were generated by the previous Transformation in a Job. The rows were passed on to this step using the "Copy rows to result" step from Transformation T.1.

First of all select "New" and "Transformation" to create the new Transformation. The first step you have to drag in the canvas is "Get rows from result" (Figure T.2.1).

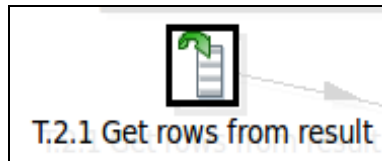


Figure T.2.1 The "Get rows from result" step

After double-clicking on the icon you can see the following menu (Figure T.2.1a). The two rows from the step "Copy rows to result" from the previous Transformation are transferred via this step to this new transformation. Just type the two fields "filename" and "path" in the table and define their type. Click "OK" to continue with the next step.

Step name

Fields :

▲	#	Fieldname	Type	Length	Precision
	1	filename	String		
	2	path	String		

Figure T.2.1a "Get rows from result" menu

### T.2.2 Replace in string

Now the already used step "Replace in string" has to be added (Figure T.2.2).

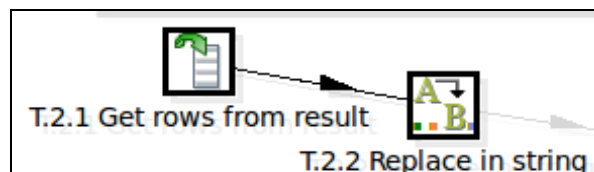


Figure T.2.2 Adding the "Replace in string" step

<sup>17</sup> <http://wiki.pentaho.com/display/EAI/Get+rows+from+result>

Figure T.2.2a shows the defined fields string of this second step. As you can see you have to choose the fields “path” and “filename” from the previous step and define the values via parameters. When clicking on “Get Fields” the two fields are transferred to this step. Now you have to define the “Search” and “Replace with” values. The search value `(/)?$` from “path” is replaced three times. First with `/${errdir}/`, second with `/${logdir}/` and last with `/${subdir}/`. Search value `/([^\/*]*)[.]gz$` from “filename” is replaced with `/${extractdir}/${1.xml}`. Do not forget to set “use RegEx” to “Y” (for Yes).

So the filenames are (compare Figure T.2):

- name of the ESE directory = path + `${subdir}`
- filename for ABCD records = filename.gz `${extractdir}/filename.xml`
- name of the ESE with errors directory = path + `${errdir}`

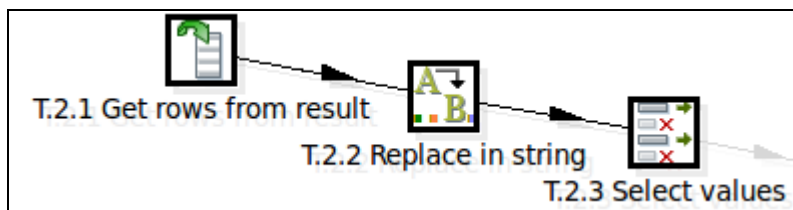
Step name: T.2.2 Replace in string								
Fields string								
#	In stream field	Out stream field	use RegEx	Search	Replace with	Replace with field	Whole Word	Case
1	path	eseErrDir	Y	<code>(/)?\$</code>	<code>/\${errdir}/</code>		N	Y
2	path	eseLogDir	Y	<code>(/)?\$</code>	<code>/\${logdir}/</code>		N	Y
3	path		Y	<code>(/)?\$</code>	<code>/\${subdir}/</code>		N	Y
4	filename		Y	<code>/([^\/*]*)[.]gz\$</code>	<code>/\${extractdir}/\${1.xml}</code>		N	Y

Figure T.2.2a Defining search and replace values in the “Replace in string” step

When this is done you can click on “OK” and go on to the next step.

### T.2.3 Select values

This step has already been used in the last Transformation. First you have to add the step to our second Transformation (see Figure T.2.3).



T.2.3 Adding the “Select values” step

In this case we use it to rename our two fields “filename” and “path” into “abcdFilename” and “eseDir” (see Figure T.2.3a). The “abcdFilename” is the name for the records in ABCD format and “eseDir” identifies the path to the ESE directory. Furthermore we select the fields “eseErrDir” and “eseLogDir” but without changing their name.

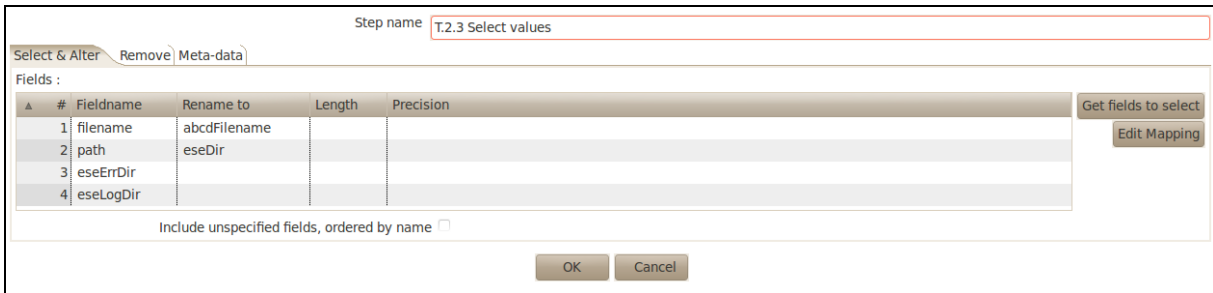


Figure T.2.3a Select and rename values

When this is done click on “OK”.

Now the last step is connected to the previous one. Again it is a “Copy rows to result” step with which we can transfer our result to the next Transformation. When this step is added the transformation should look like the one in *Figure T.2* and is finished. Also for this Transformation there is no preview possible (compare Figure T.1.8a).

Now it is time for the last Transformation.

### T.3 The “ABCD206\_to\_ESE3” Transformation

Figure J.3 gives again an overview of the whole Job. The Transformation we are going to create now is marked red.

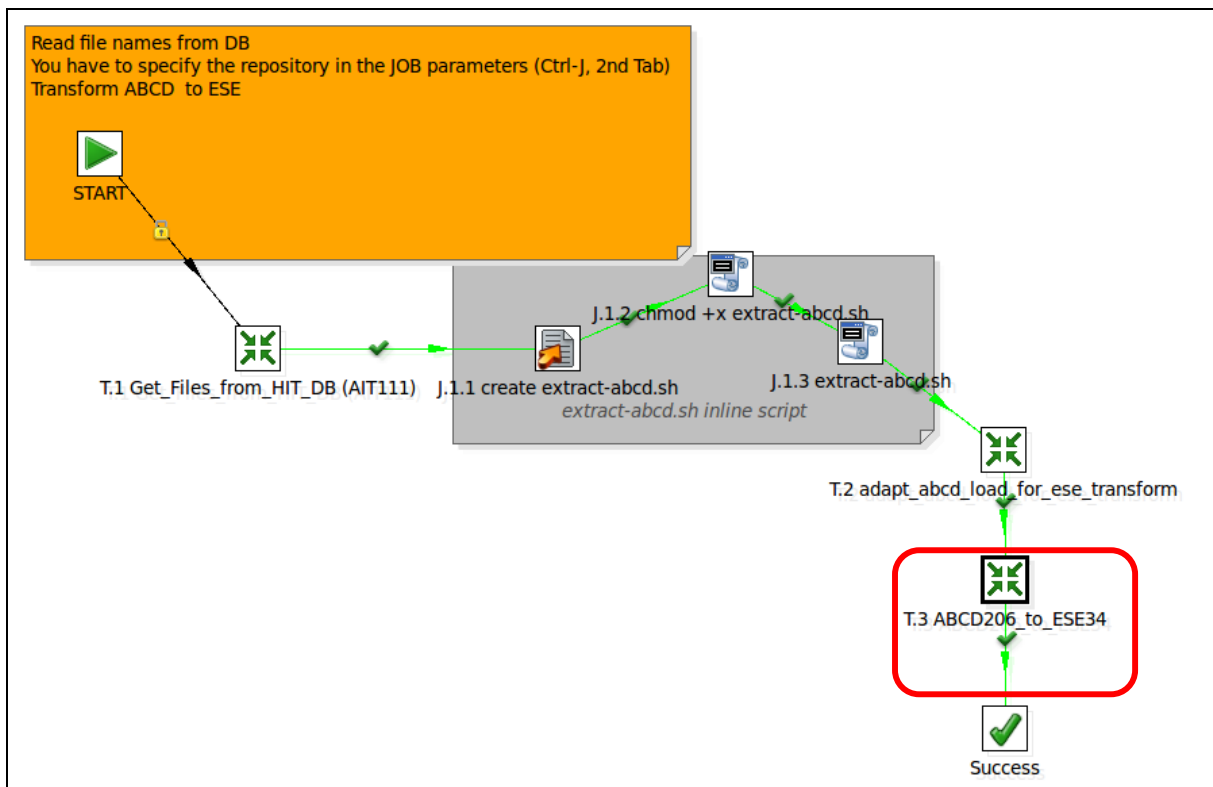


Figure J.3 The Job with the current Transformation marked red

Figure T.3 shows the whole Transformation “ABCD206\_to\_ESE34”. This Transformation uses an XSL Stylesheet to convert ABCD to ESE files. It includes a filter step too, which sorts out records, that do not follow the needed standard.

At the end the ESE records are copied to the result set and a log file is created.

Every step will be explained in the next sections.

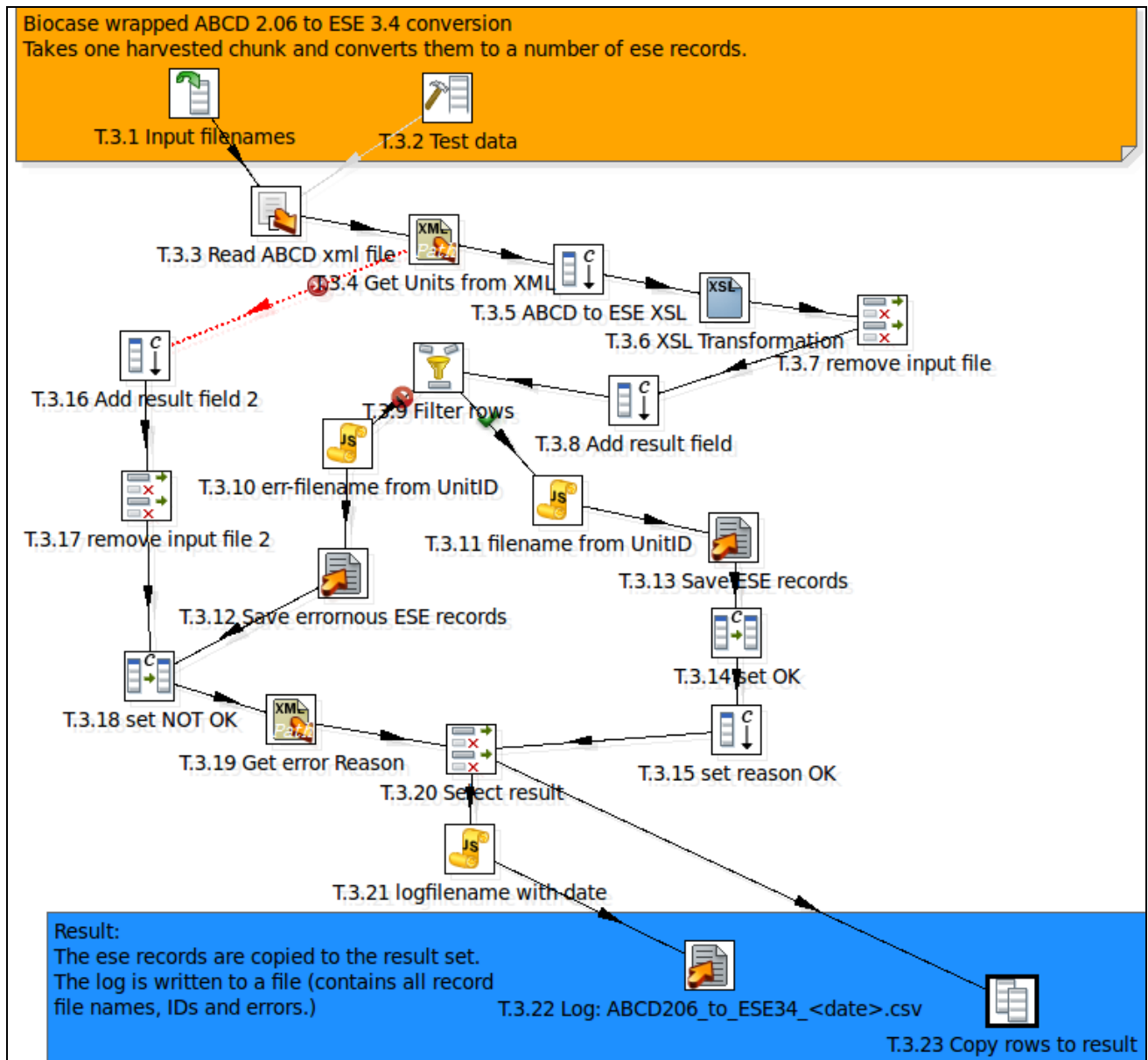


Figure T.3 The ABCD206\_to\_ESE34 Transformation

### T.3.1 Get rows from result – Input filenames

As you can see there are two input steps (Figure T.3.1). First we will have a look on the “Get rows from result” step which is named “Input filenames” in our example. Like done before, we copy our result from

the previous Transformation to this one. When double-clicking on the icon, you can choose the fields by typing them under "Fieldname" (Figure T.3.1a).

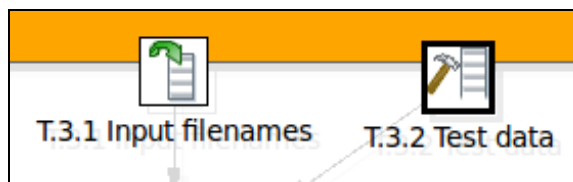


Figure T.3.1 The two input steps "Input filenames" and "Test data"

You can see that the two fields from the last transformation have to be transferred into the next one. Note that you have to use the new names you created in the Transformation before.

Step name:

Fields :

▲	#	Fieldname	Type	Length	Precision
	1	abcdFilename	String		
	2	eseDir	String		
	3	eseErrDir	String		
	4	eseLogDir	String		

OK Cancel

Figure T.3.1a Chosen fields in the "Get rows from result" step

When you have done that, click "OK" and continue with the second input.

### T.3.2 Generate rows<sup>18</sup> – Test data

With this step it is possible to create a number of empty or equal rows. In our case it is used to add the fields "abcdFilename", "eseDir", "eseErrDir" and "eseLogDir" with the right path to the directories or the files.

<sup>18</sup> <http://wiki.pentaho.com/display/EAI/Generate+Rows>

Step name: T.3.2 Test data  
Limit: 1

Fields:

#	Name	Type	Forr	Len	Prec	Cui	Deci	Group	Value
1	abcdFilename	String							/opt/hit/2545906-bgbm-herbar/herbarium_berolinense/extracted/search_response.5235.xml
2	eseDir	String							/opt/hit/2545906-bgbm-herbar/herbarium_berolinense/ese/
3	eseErrDir	String							/opt/hit/2545906-bgbm-herbar/herbarium_berolinense/eseWithErrors/
4	eseLogDir	String							/opt/hit/2545906-bgbm-herbar/herbarium_berolinense/log/

Buttons: OK, Preview, Cancel

Figure T.3.2a Generated rows in the menu of the "Generate rows" step

The values contain the directories where the needed data is stored and therefore yours will differ from the one shown in Figure T.3.2a except the values in bold beneath. This directory and filename pattern has been created with the Transformations before.

Value for abcdFilename: /opt/hit/2545906-bgbm-herbar/herbarium\_berolinense/**extracted/search\_response.000.xml**

Value for eseDir: /opt/hit/2545906-bgbm-herbar/herbarium\_berolinense /**ese/**

Value for eseErrDir: /opt/hit/2545906-bgbm-herbar/herbarium\_berolinense/**eseWithErrors/**

Value for eseLogDir: /opt/hit/2545906-bgbm-herbar/herbarium\_berolinense/**log/**

When this is done click "OK" and move on to the next step.

### T.3.3 Load file content in memory – Read ABCD xml file

Figure T.3.3 shows how the two input steps are connected with the "Load file content in memory" step. Its name reveals its use: loading the content of files and "store" it in memory. We use it to read XML files with the ABCD schema.

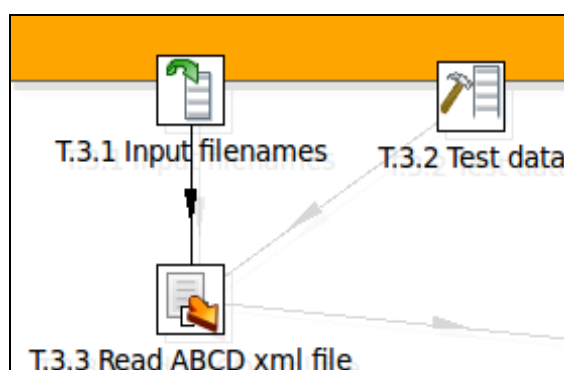


Figure T.3.3 Adding the step "Load file content in memory"

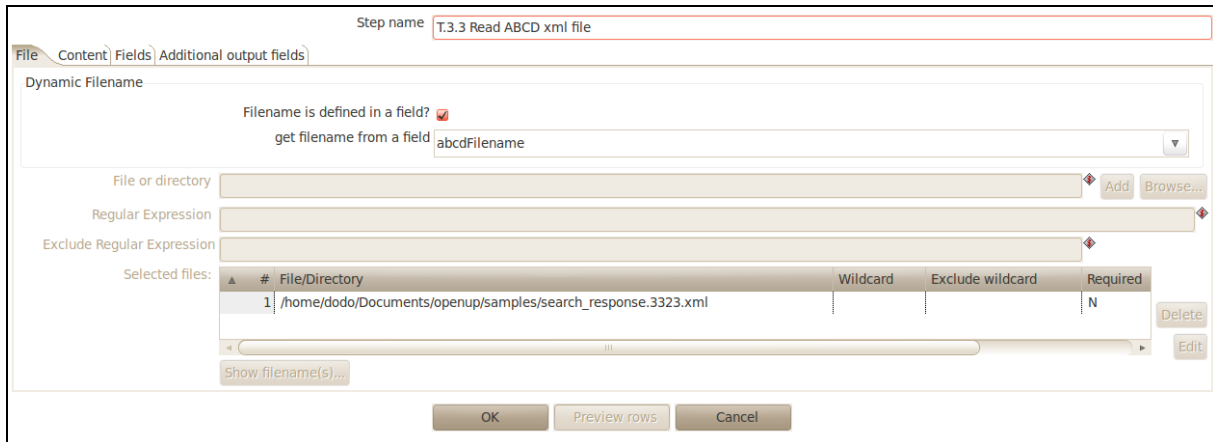


Figure T.3.3a File section of the “Load file content in memory” step

As you can see in *Figure T.3.3a* this step contains four different sections: File, Content, Fields and Additional output fields. First in the File section you have to choose one test XML file which has to be added to “Selected files” via the buttons “Browse” and “Add”. Again the path shown in *Figure T.3.3a* will differ from your own. Furthermore make sure the “Filename is defined in a field” option is activated. The “abcdFilename” field is then selected after “get filename from a field” because this field contains the filename.

**NOTE:** If the “Filename is defined in a field” option is activated before adding the file, choosing input is not possible.

When this is done you can move on to the next section – Fields (*Figure T.3.3b*). Both sections Content and Additional output fields are not necessary in this case.

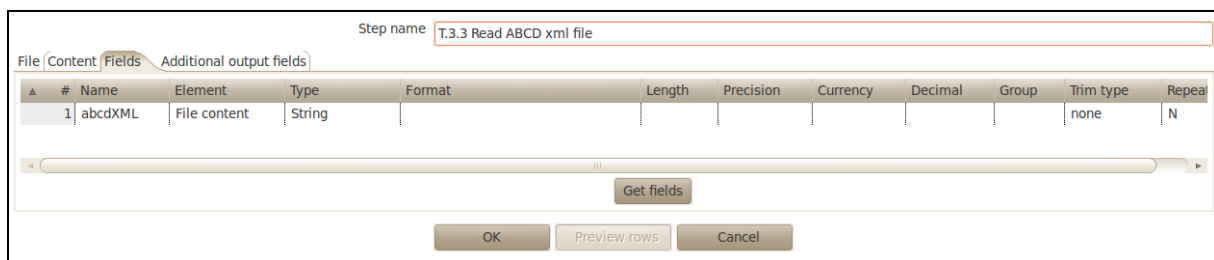


Figure T.3.3b Fields section of “Load file content in memory”

Figure T.3.3b shows the field “abcdXML” from the element “File content”. Do not use the “Get fields” button, but type the field name and choose “File content” via drop down in the column “Element”. When this is done click “OK”.

### T.3.4 Get XML data<sup>19</sup> – Get Units from XML

The next step that needs to be added is the “Get XML data” step. In our example it is called “Get Units from XML”.

After adding this step the transformation should look similar to the one shown in *Figure T.3.4*.

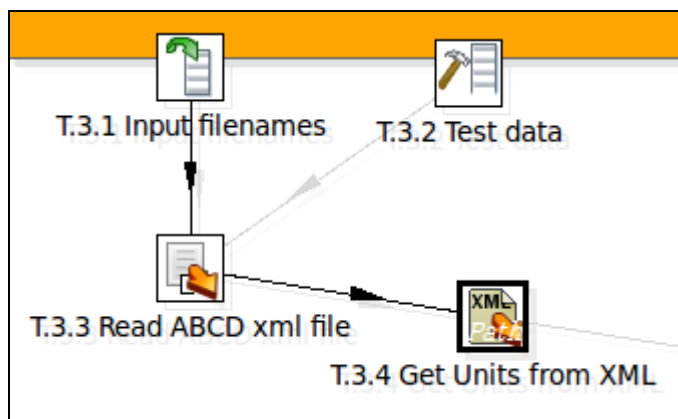


Figure T.3.4 Adding the “Get XML data” step

This step is used to get data from one or more XML files by using XPath. When double-clicking on the icon you can see the following window (*Figure T.3.4a*):

In the File section it is important to choose “XML source is defined in a field” and “get XML source from a field”. The selected field is “abcdXML” which we created in the previous step.

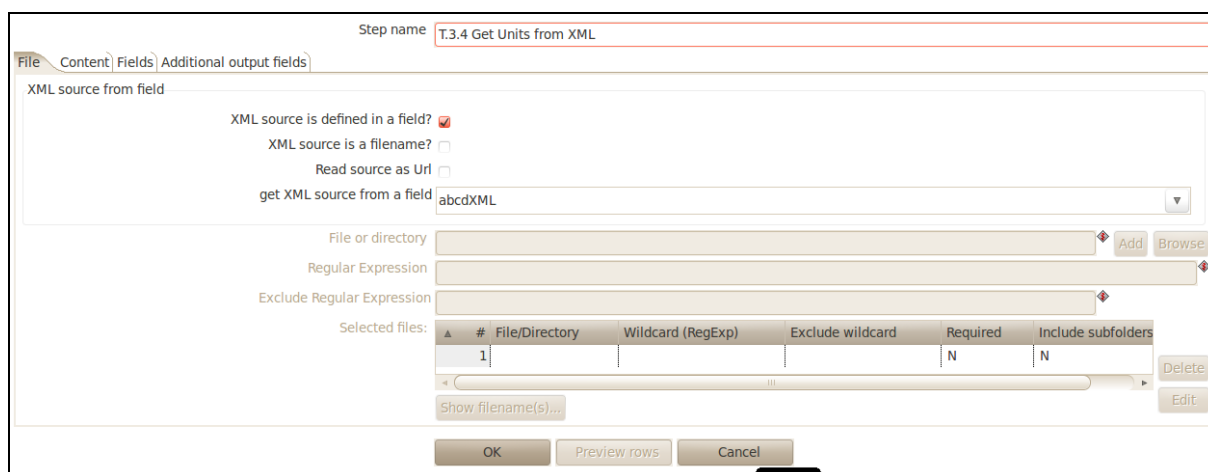


Figure T.3.4a File section of “Get XML data”

<sup>19</sup> <http://wiki.pentaho.com/display/EAI/Get+Data+From+XML>



In the Content section (see Figure T.3.4b) you can choose the XPath<sup>20</sup> of your ABCD record. In this case the XPath defines the repeating element of the XML document. Because we need the information in the element "Unit" the XPath statement is:

**/biocase:responses/biocase:content/abcd:DataSets/abcd:DataSet/abcd:Units/abcd:Unit**

You can either type the statement or select it by clicking on "Get XPath nodes". The option "Do not raise an error if no files" must be activated.

Figure T.3.4b Content section of "Get XML data"

Finally go to the Fields section (see Figure T.3.4c) and click on "Get fields". You can delete every field except "SourceInstitutionID", "SourceID" and "UnitID". These three values will later be the unique identifier of the new ESE record.

#	Name	XPath	Element	Result type	Type	Format	Length	Precision	Currency	Decimal
1	SourceInstitutionID	abcd:SourceInstitutionID	Node	Value of	String					
2	SourceID	abcd:SourceID	Node	Value of	String					
3	UnitID	abcd:UnitID	Node	Value of	String					

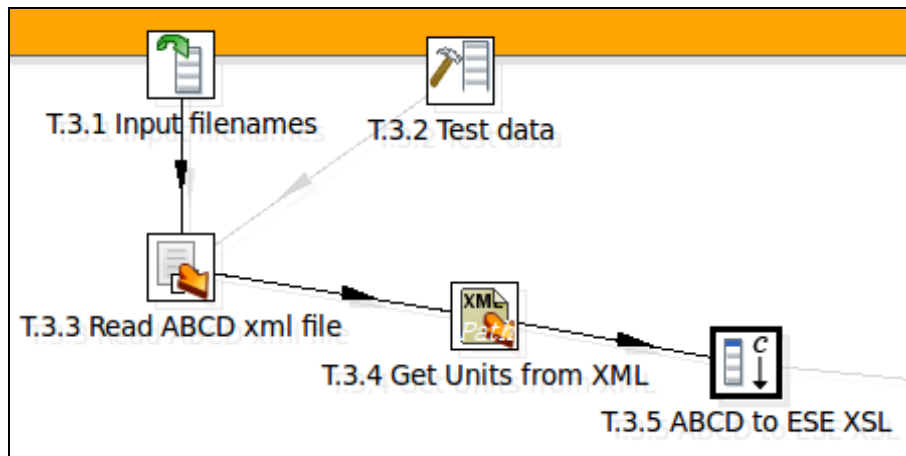
Figure T.3.4c Fields section of "Get XML data"

When you have chosen these three fields, make sure you define "Element", "Result type" and "Type" via drop down menu. When this is done click "OK". Additional output fields are not needed in this example.

<sup>20</sup> If you need further information on XPath please visit <http://www.w3schools.com/xpath/default.asp>

### T.3.5 Add constants – ABCD to ESE XSL

You already got to know the “Add constants” step in previous chapters. In this Transformation we use the step to add an XSL stylesheet<sup>21</sup> which transforms one XML document into another. First of all add the step to the Transformation (*Figure T.3.5*).



*Figure T.3.5 Adding the “Add constants” step*

After double-clicking on the icon you can define your constant (*Figure T.3.5a*). You have to fill in the whole XSL file in the field “Value” of the first row.

The name of this field is “abcd2EuropeanaXSL” in our example. You can choose the name by yourself but make sure the Type is defined as String.

Please note: The XSL Stylesheet is described in Part 2 of this document.

<sup>21</sup> <http://en.wikipedia.org/wiki/XSL>

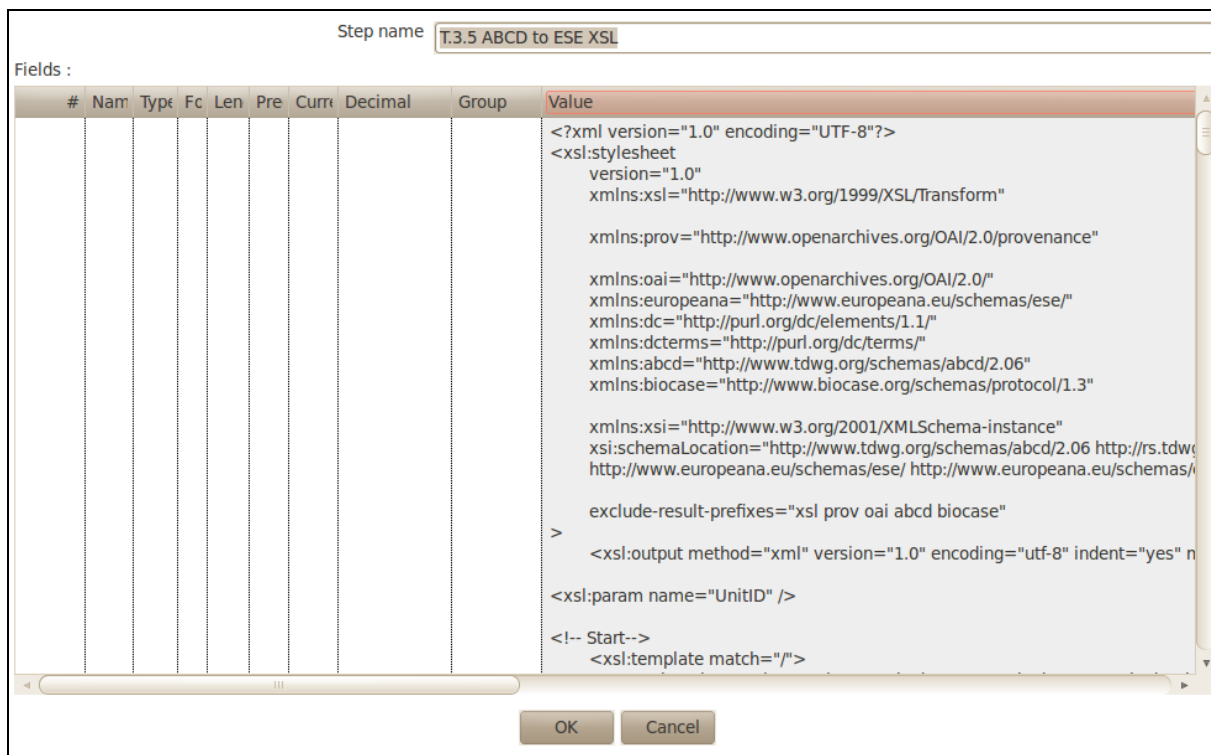


Figure T.3.5a Value of the "Add constants" step

### T.3.6 XSL Transformation<sup>22</sup>

After adding the "XSL Transformation" step (see Figure T.3.6) you can define the settings of this step (Figure T.3.6a).

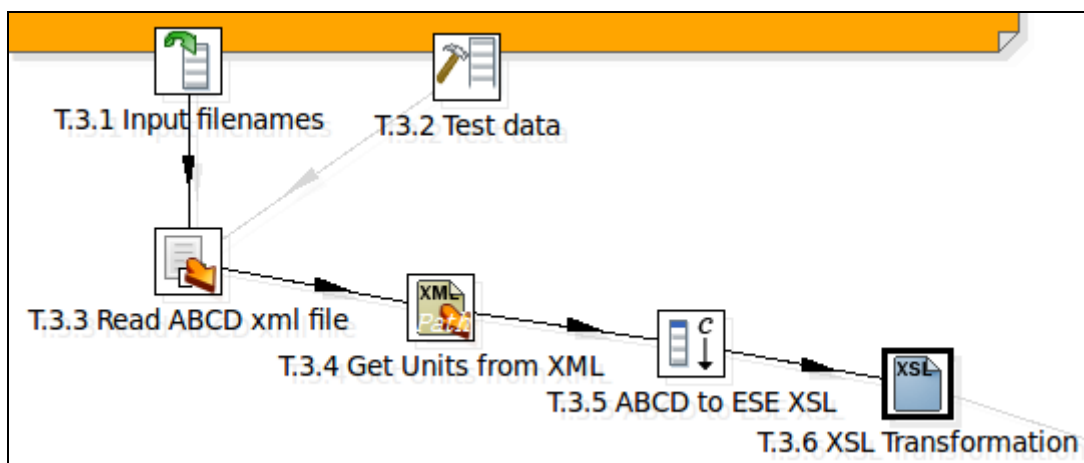


Figure T.3.6 Adding the "XSL Transformation" step

<sup>22</sup> <http://wiki.pentaho.com/display/EAI/XSL+Transformation>

In “Settings” you have to define the XML field name (abcdXML) and the result field name (eseXML). “abcdXML” can be selected via drop down menu. the “eseXML” field has to be typed in. It is the name of the result field of the XSL Transformation. The “XSL source defined in a field” option has to be activated and this “XSL file name field” has to be determined. It is the constant value we defined the step before: “abcd2EuropeanaXSL” which contains our stylesheet. Finally choose the “XSLT Factory”: “SAXON”.

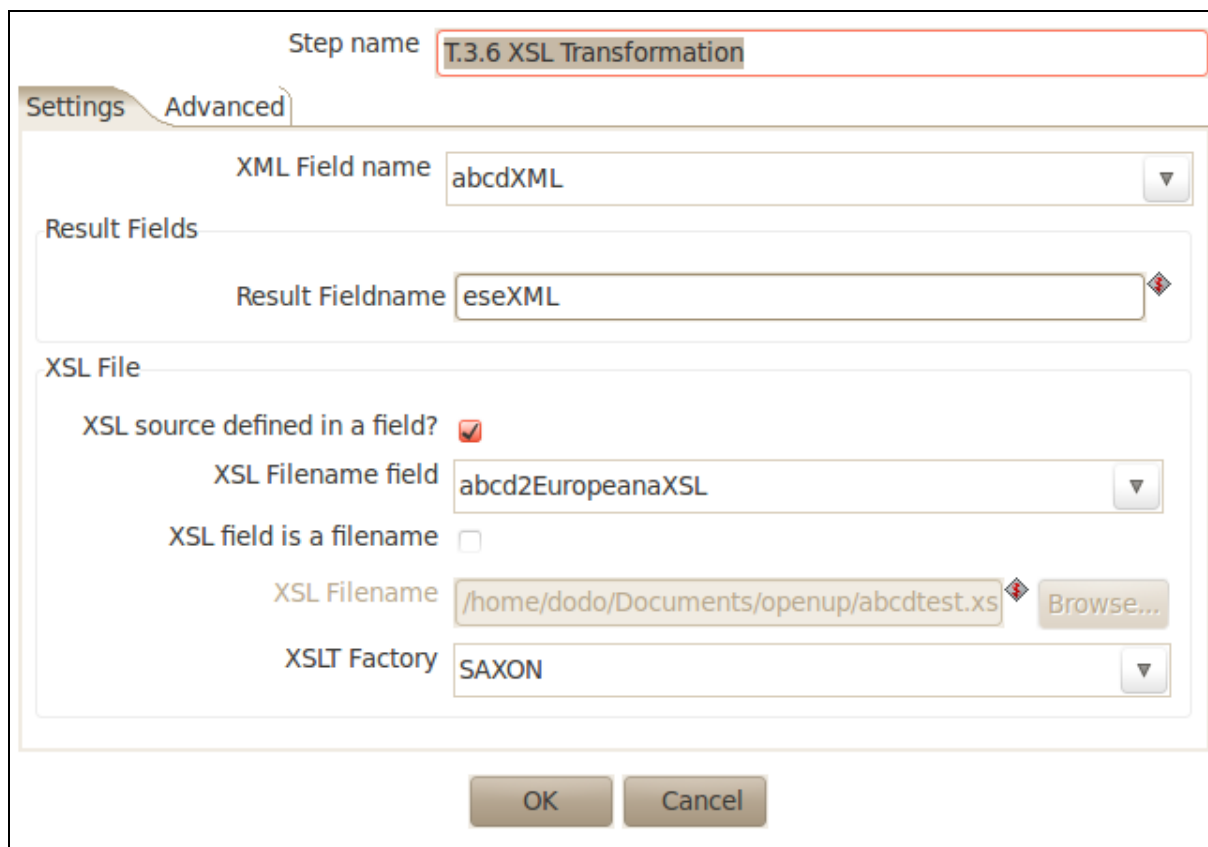


Figure T.3.6a Settings of the XSL Transformation

After that you can go on to the “Advanced” tab. Here you have to fill in the three fields “SourceInstitutionID”, “SourceID” and “UnitID” you have chosen in the “Get XML data” step before (see Figure T.3.6b).

When clicking on “Get Fields” every field from previous steps are selected. You can either use this option and delete the unnecessary fields or choose the three needed fields with the drop down menu.

Step name

Settings **Advanced**

Output properties

#	Property name	Property value
1		
2		
3		

Parameters

#	Stream Field	Parameter name
1	SourceInstitutionID	SourceInstitutionID
2	SourceID	SourceID
3	UnitID	UnitID

Get fields

OK Cancel

Figure T.3.6b The Advanced section of XSL Transformation

When everything is filled in correctly click "OK" and move on to the next step.

### T.3.7 Select values – remove input file

Now you can add another "Select values" step. In our example it is called "remove input file". It is the seventh step in our transformation (see Figure T.3.7).

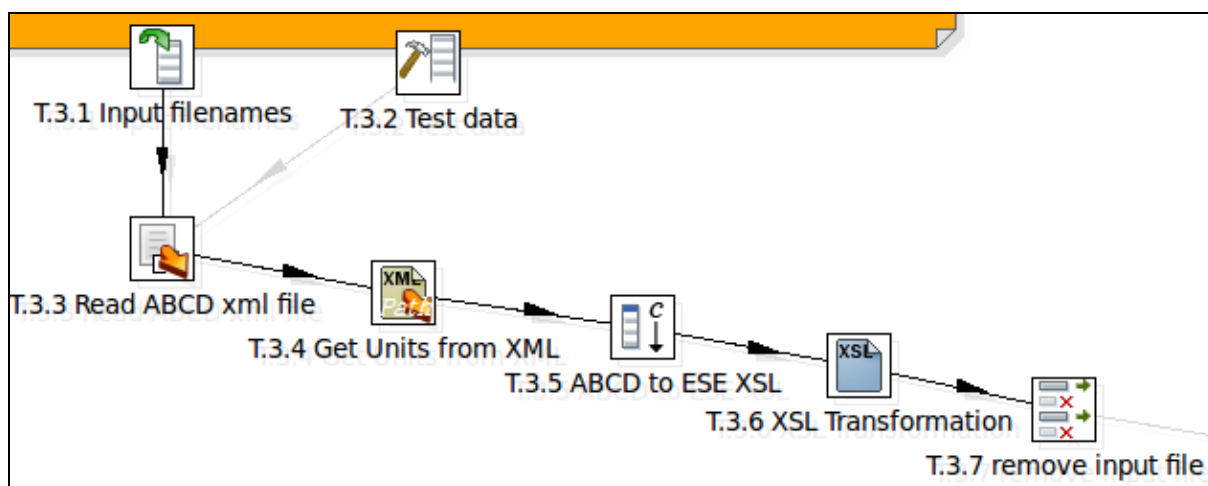
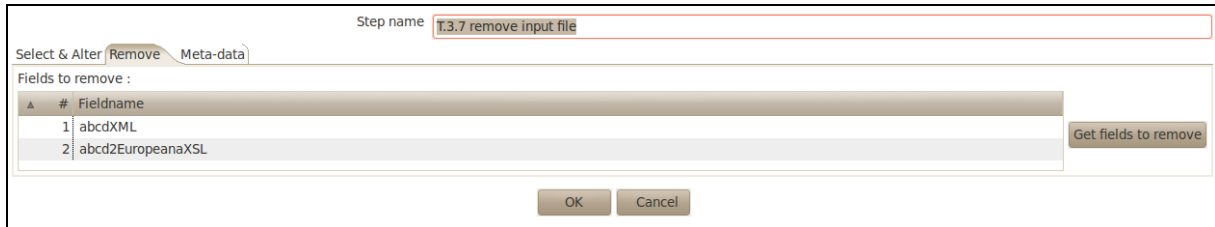


Figure T.3.7 Adding another "Select values" step

This step is used to remove the fields “abcdXML” (from the “load file content in memory” step) and “abcd2EuropeanaXSL” (see *Figure T.3.7a*). Make sure you choose the fields in the Remove section. Again you can either click on “Get fields to remove” or select them by hand.

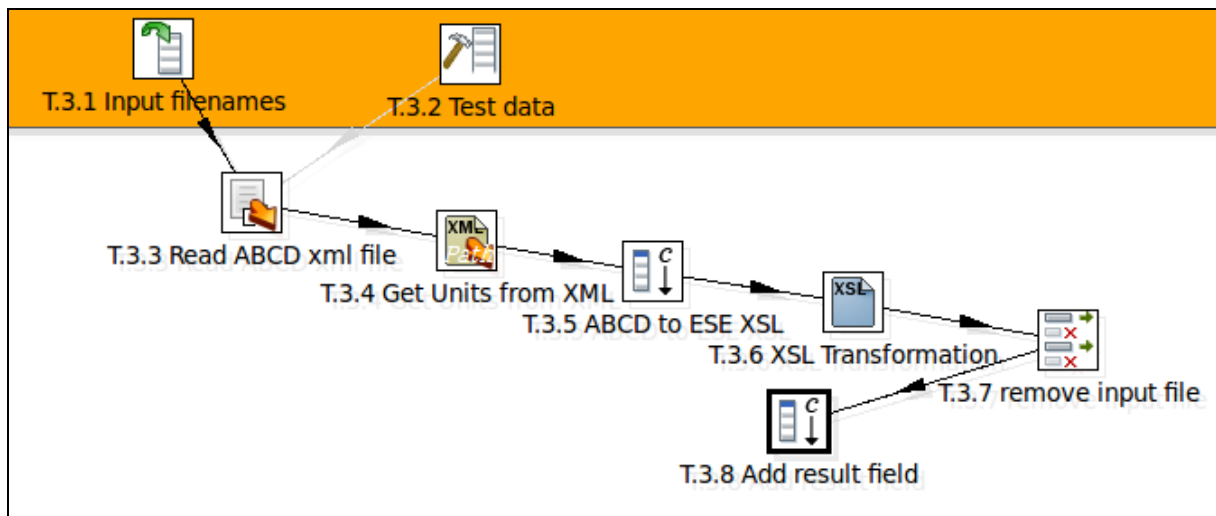


*Figure T.3.7a “Select values” step – removing “abcdXML” and “abcd2EuropeanaXSL”*

After clicking “OK” it is time for the next step.

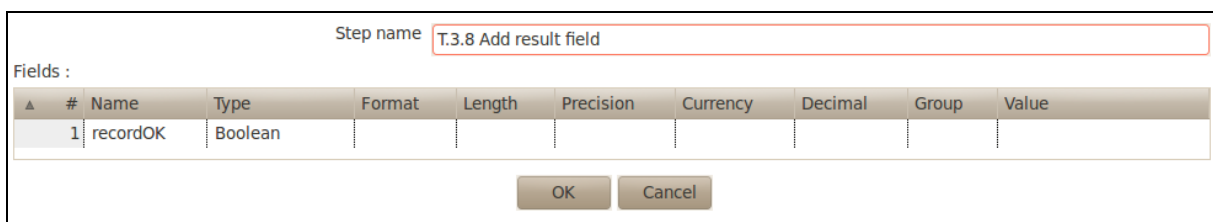
### T.3.8 Add constants – Add result field

Now you have to add another “Add constants” step. In our example the step is called “add result field (see *Figure T.3.8*). We need this field to determine whether a record is correct or not.



*Figure T.3.8 Adding another “Add constants” step*

Figure *T.3.8a* shows the result field called “recordOK”. The type you have to select for this field is “Boolean”.



*Figure T.3.8a Adding the result field “recordOK”*

### T.3.9 Filter Rows<sup>23</sup>

The “Filter Rows” step is a special step because you need two output fields. *Figure T.3.9* shows our Transformation. Note that the Filter Rows step has two output steps (T.3.10 and T.3.11).

This step filters rows by using simple equations. If an ESE record does not fit the necessary standard it goes the left way (“T.3.10 err-filename from UnitID”), otherwise it goes to the step “T.3.11 filename from UnitID” on the right. The paths are also defined through the symbols on the hops.

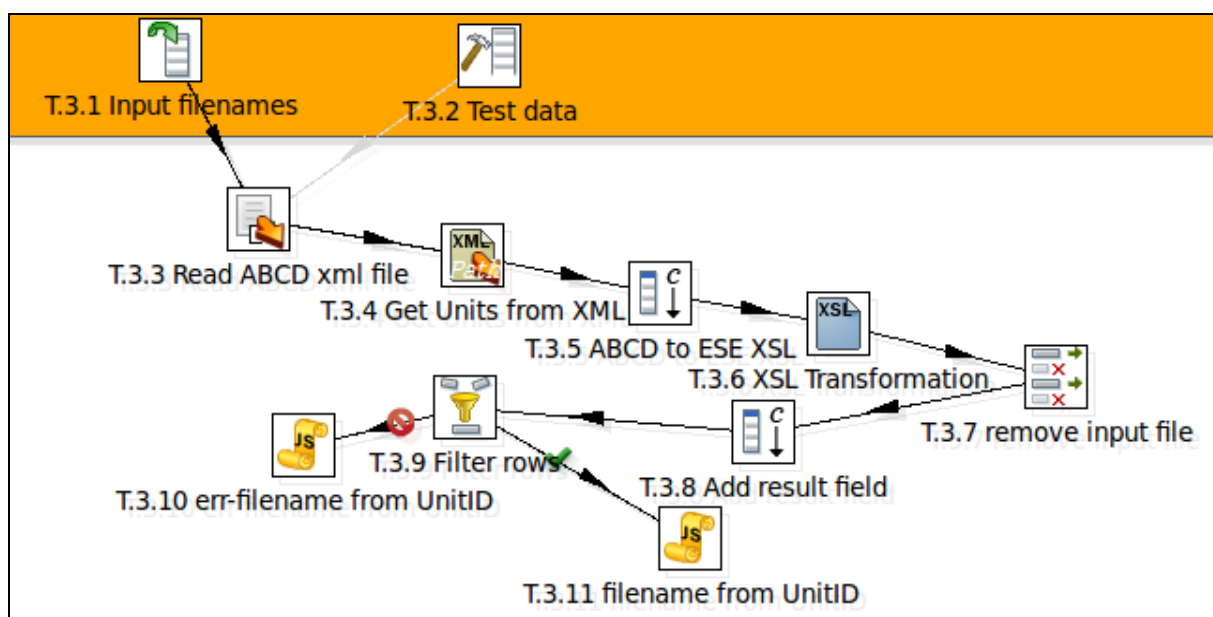


Figure T.3.9 “Filter rows” step with two output steps

When you want to connect the “Filter rows” step with other steps via hop you have to choose “Result is TRUE” (with T.3.11) or “Result is FALSE” (with T.3.10). When double-clicking on the icon you can define your output steps too.

Figure T.3.9a shows how the conditions for the “Filter rows” step are defined. You can see there the two rows “Send ‘true’ data to step” and “Send ‘false’ data to step”. You just have to select the next steps (seen in *Figure T.3.9*). After that you can create a condition. The condition in our example is looking for the value “\_record\_invalid ‘true’”. When it occurs in “eseXML” it is a record which is not correct and therefore sent to the “err-filename from UnitID” step.

<sup>23</sup> <http://wiki.pentaho.com/display/EAI/Filter+rows>

Figure T.3.9a The “Filter rows” step with the condition

### T.3.10 Modified Java Script Value<sup>24</sup> – err-filename from UnitID

After the “Filter Rows” step there are two paths in our Transformation: one that processes the correct files, and one to process wrong records (compare Figure T.3.9).

In step T.3.10 we are using JavaScript to get filenames – but in this case the names of the incorrect files. This step is used for building Java Script expressions. After double-clicking on the icon you can start to script (Figure T.3.10).

Δ	#	Fieldname	Rename to	Type	Length	Precision	Replace value 'Fieldname' or 'Rename to'
	1	filename		String			N
	2	fileid		String			N

Figure T.3.10 Getting the error filename from UnitID with JavaScript

<sup>24</sup> <http://wiki.pentaho.com/display/EAI/Modified+Java+Script+Value>



In this step you have two inputs: the Java script and the Fields. The Java script<sup>25</sup> looks as follows:

```
var sid = encodeURI(SourceID.replace(/ /g, "_"));
var siid = encodeURI(SourceInstitutionID.replace(/ /g, "_"));
var uid = encodeURI(UnitID.replace(/ /g, "_"));

var fileid = sid + "-" + siid + "-" + uid;
var r = new Number(Math.floor(Math.random() * 256*256)).toString(16);
var filename = eseErrDir + "/" + fileid + "~" + r + ".xml"
```

Note that we choose the filename from the field "eseErrDir" because we are working with the incorrect records.

After scripting you must not forget to choose the two fields "filename" and "fileid" with their type "String". This was the step to get the filenames from incorrect records. Now we have a look at the correct files.

### ***T.3.11 Modified Java Script value – filename from UnitID***

After adding a second "Modified Java Script value" step (under the Scripting directory) the Transformation should look like the one shown in *Figure T.3.9*.

In our example the Java Script is used to get the filename of correct files from the field "UnitID" and reads as follows(see Figure T.3.11):

```
var sid = encodeURI(SourceID.replace(/ /g, "_"));
var siid = encodeURI(SourceInstitutionID.replace(/ /g, "_"));
var uid = encodeURI(UnitID.replace(/ /g, "_"));

var fileid = sid + "-" + siid + "-" + uid;
var r = new Number(Math.floor(Math.random() * 256*256)).toString(16);
var filename = eseDir + "/" + fileid + "~" + r + ".xml"
```

In contrary to the last step we get the filename from the field "eseDir" because we need the filenames from the correct records.

Do not forget to choose "filename" and "fileid" in the Fields section. When this is done it is time to move on with the Transformation.

---

<sup>25</sup> If you need further information on JavaScript please visit <http://www.w3schools.com/js/default.asp>



Figure T.3.11 Getting the error filename from UnitID with JavaScript

### T.3.12 Text file output<sup>26</sup> – Save erroneous ESE records

As you can see in Figure T.3.12 “Text file output” step called “Save erroneous ESE records” is one of the output steps of the “Filter rows step”. Figure T.3.12a shows how the step has to be defined to save the incorrect files.

<sup>26</sup> <http://wiki.pentaho.com/display/EAI/Text+File+Output>

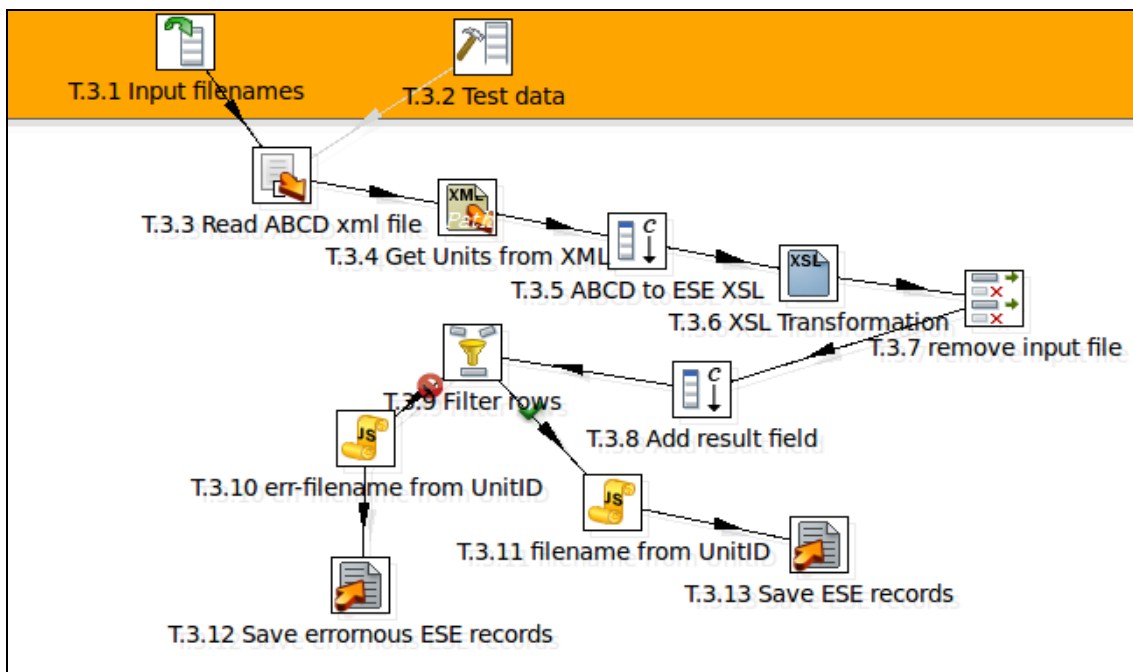


Figure T.3.12 Adding two "Text file output" steps

In the file section you have to choose "Accept file name from field?" and select the field "filename" beneath because it contains the names of the files.

Figure T.3.12a File section of "Save erroneous ESE records"

When this is done move on to the Content section (see Figure T.3.12b).

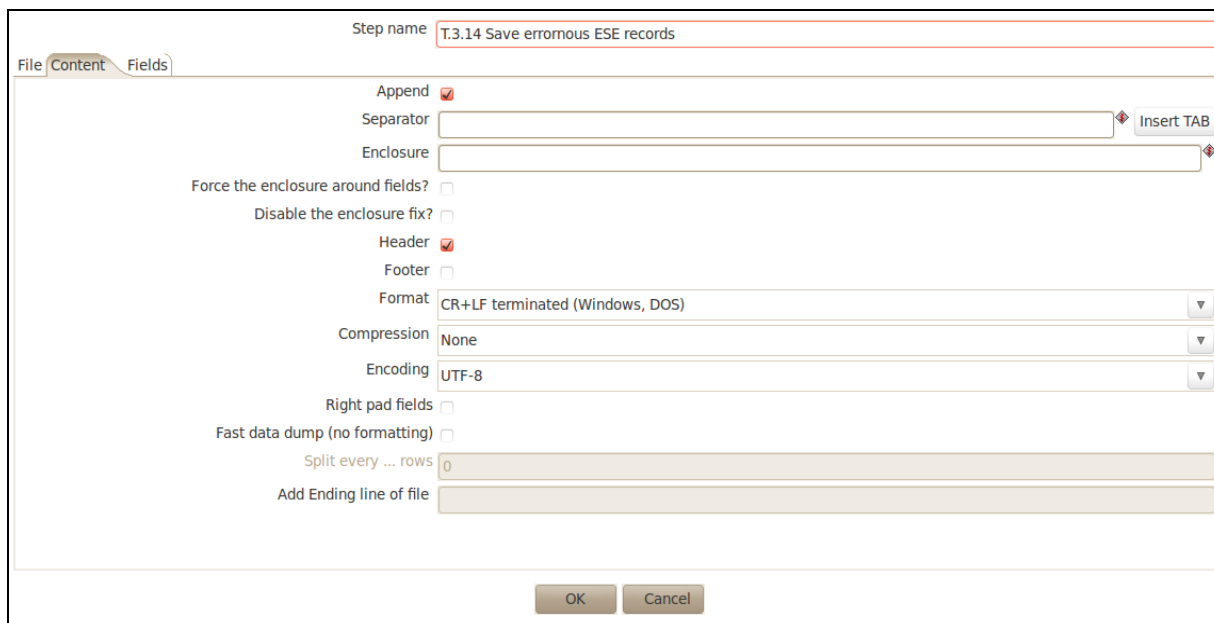
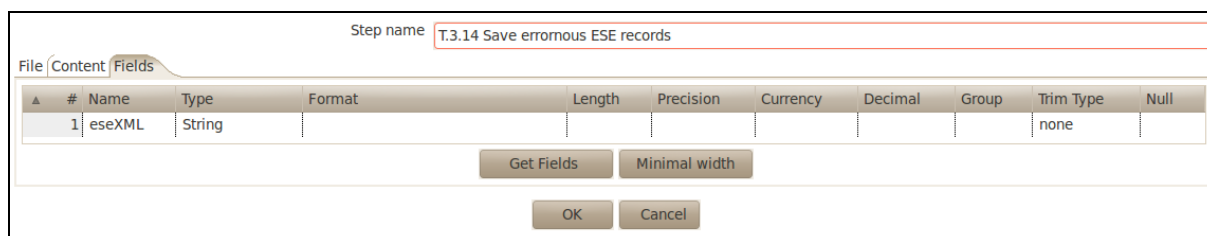


Figure T.3.12b Content section of "Save erroneous ESE records"

You can see that the option "Append" is selected - it appends lines to the end of the specified file. Moreover a "Header" has been chosen. "Format", "Compression" and "Encoding" are default values. Now only the Fields section is missing (see Figure T.3.12c). Only one field has to be chosen: "eseXML" because this field contains the ESE record.



#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim Type	Null
1	eseXML	String							none	

Figure T.3.12c Fields section of "Save erroneous ESE records"

When this is done click "OK" and go on with the next step.

### T.3.13 Text file output – save ESE records

We have defined the way to save incorrect files in the previous chapter now it is time to save the correct ESE records. To achieve this a "Text file output" step is added to T.3.11 (see Figure T.3.12). The settings are very similar to T.3.12. First we will have a look on the File section. (T.3.13a)

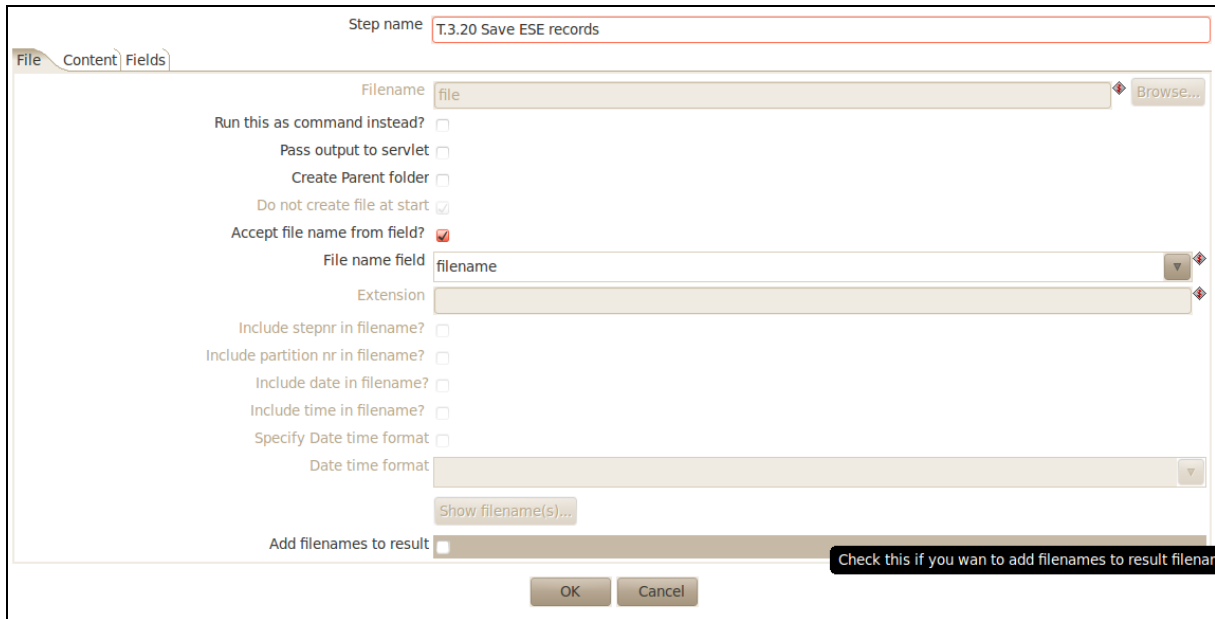


Figure T.3.13a File section of "Save ESE records"

As you can see select the option "Accept file name from field?" and choose the field "filename" beneath. When this is done move on to the Content section (see Figure T.3.13b).

Here a "Header" is chosen. The other values are selected by default.

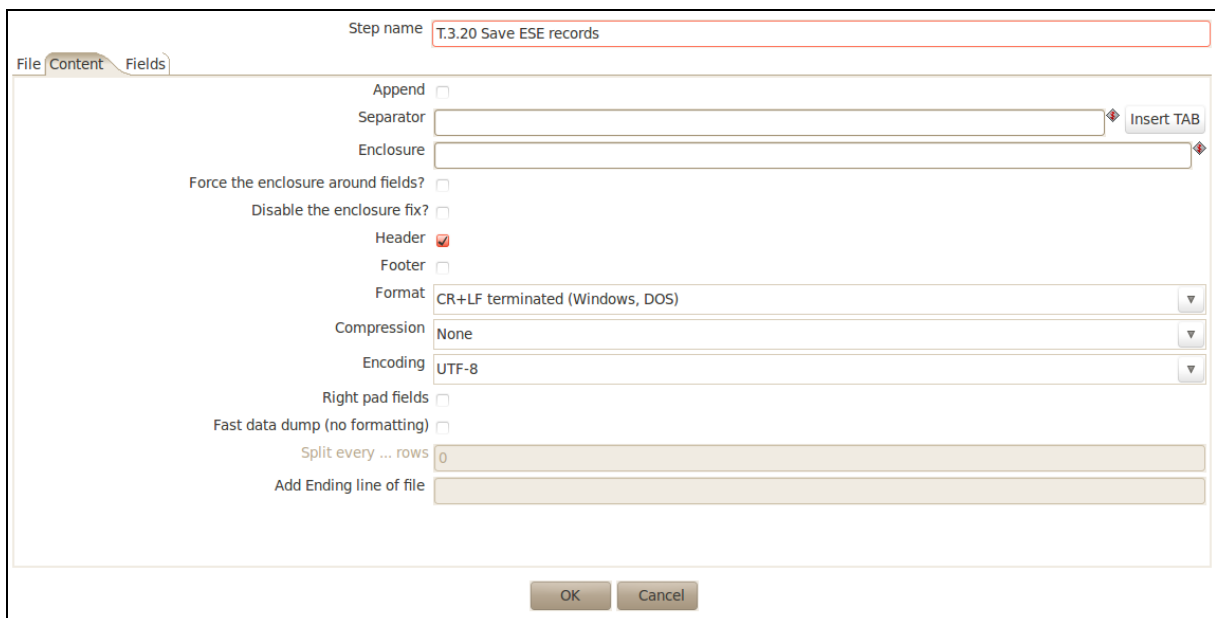


Figure T.3.13b Content section of "Save ESE records" with default values

In the File section (see Figure T.3.13c) the field "eseXML" with its type "String" has to be selected. When finished click "OK".

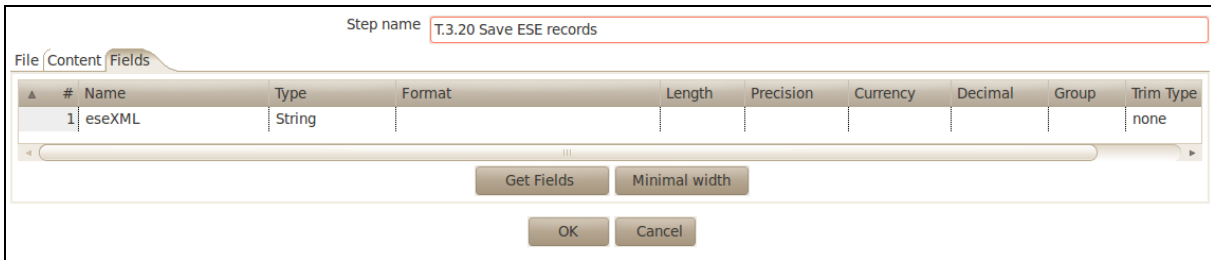


Figure T.3.13c Fields section of "Save ESE records"

### T.3.14 Set field value to constant – set OK

Now we have to add a "Set field value to constant" step (see Figure T.3.14). Please note that this step is added to the path where the correct files go. So the field "recordOK" needs to have another value to tell us that the record is right: "Y" for Yes (see Figure T.3.14a).

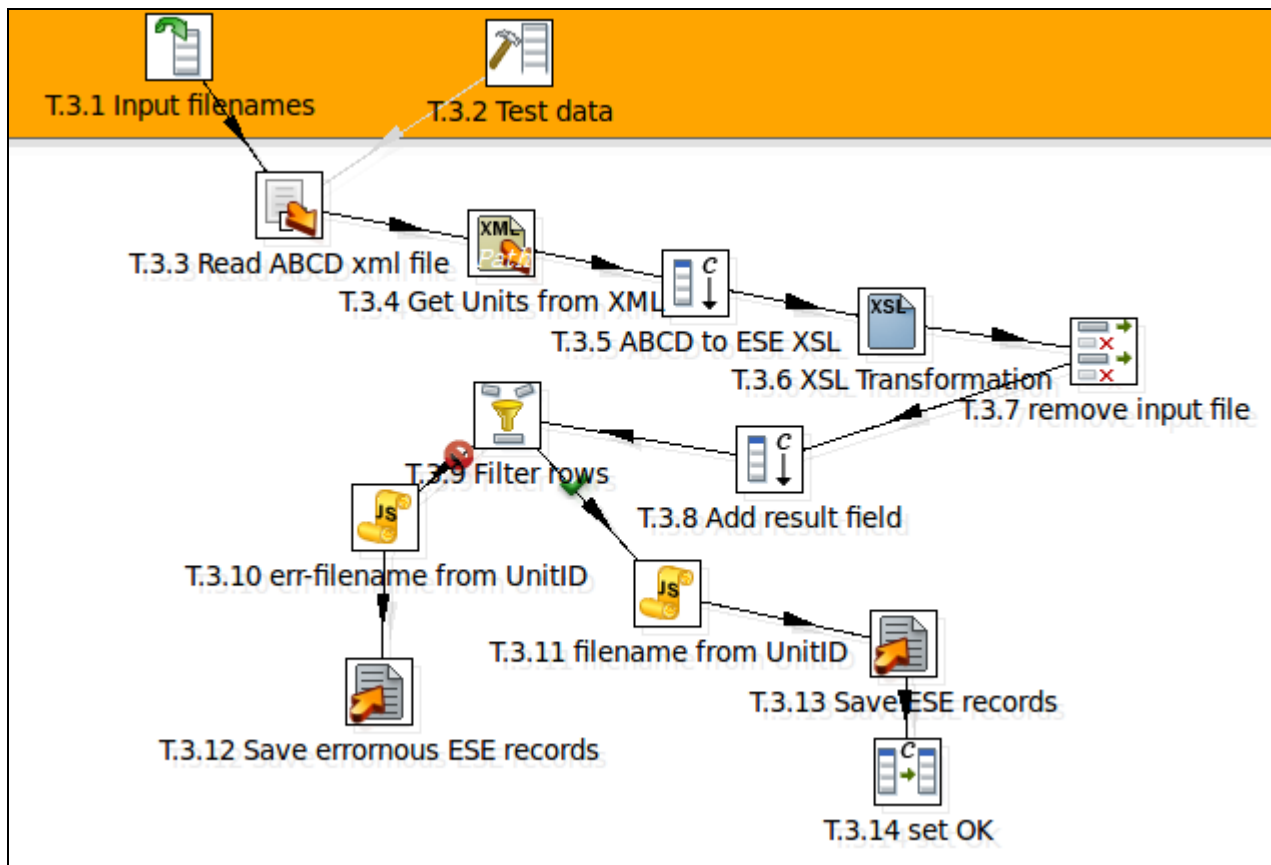


Figure T.3.14 Adding a "Set field value to constant" step to our Transformation

Step name

Use variable in constant

Fields

#	Field	Replace by value	Conversion mask (Date)
1	recordOK	Y	

OK    Get Fields    Cancel

Figure T.3.14a Defining the constant "Y"

### T.3.15 Add constant value –set reason OK

Now we are adding an "Add constant value" step (see Figure T.3.15) to define the value for "errReason" when the records are correct. As you can see in Figure T.3.15a the value chosen is "OK".

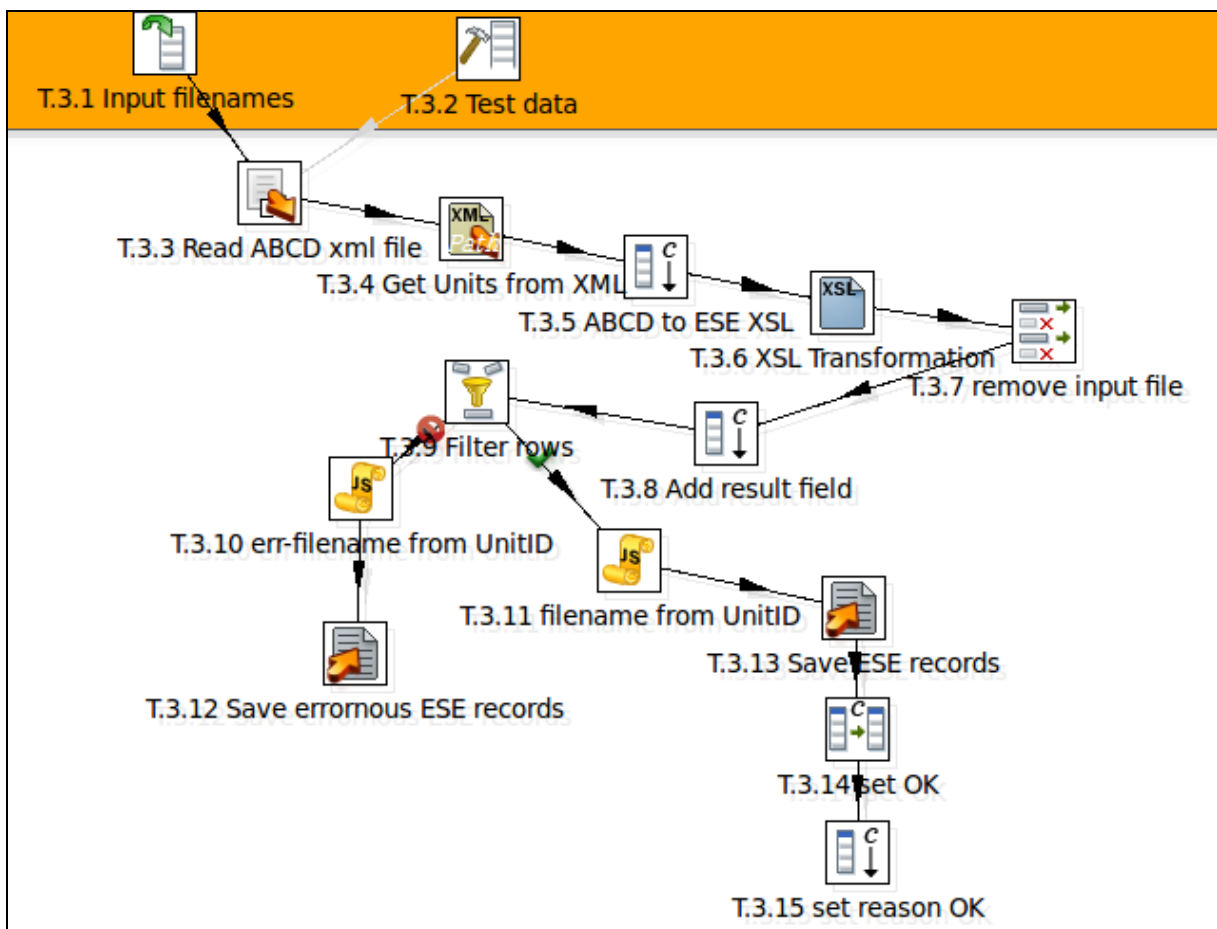


Figure T.3.15 Adding another "Add constant value" step

Step name

Fields :

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value
1	errorReason	String							OK

OK Cancel

Figure T.3.15a Setting errorReason "OK"

**T.3.16 Add constants<sup>27</sup> – Add result field 2**

Now it is time to find a way sorting out XML files in ABCD schema that do not have Units – and therefore no input. To achieve this add another "Add constants" step to your Transformation. As you can see in Figure T.3.16 the step "Get Units from XML" (T.3.4) is connected via hop with the new field. When connecting you are asked to choose "Main output of step" or "Error handling of step". You have to choose "Error handling of step" and the hop looks like in Figure T.3.16.

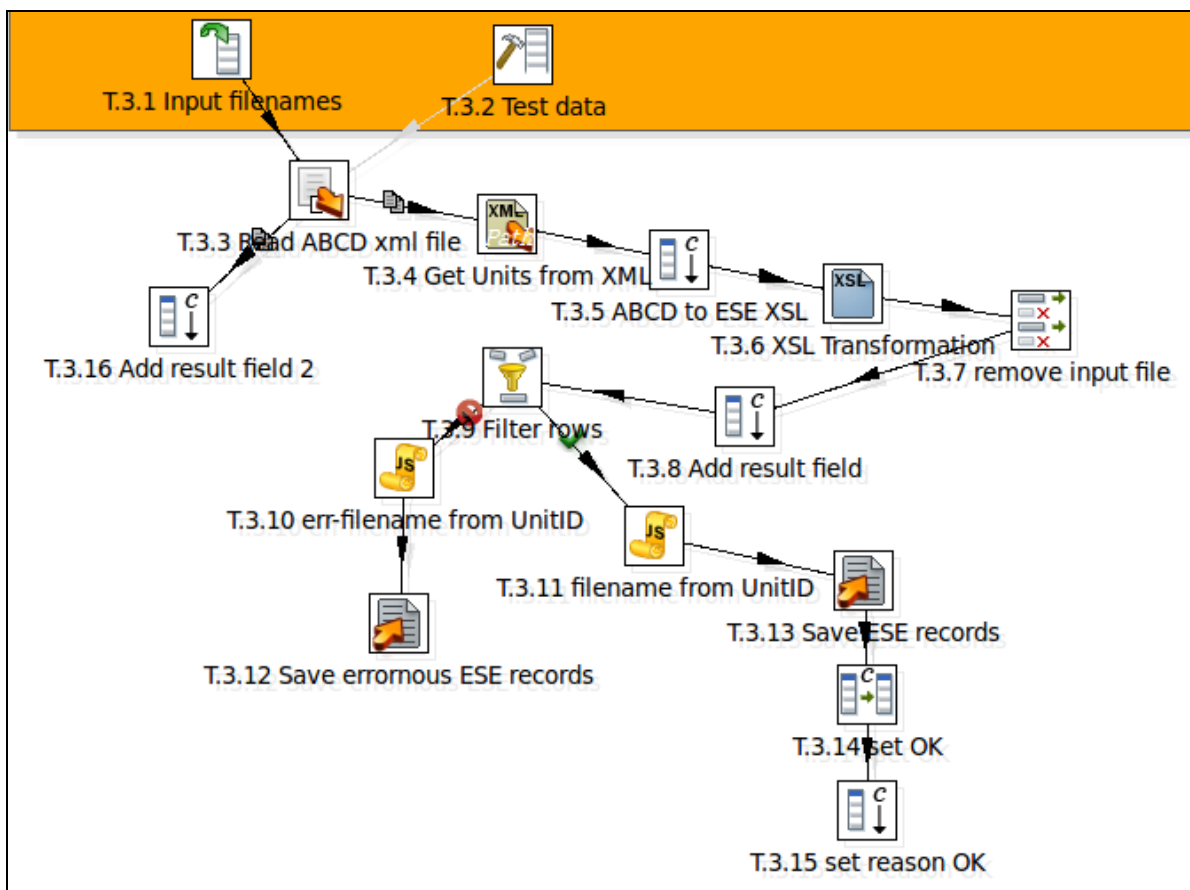


Figure T.3.16 Adding an "Add constants" step to T.3.3

<sup>27</sup> <http://wiki.pentaho.com/display/EAI/Add+Constants>



In this step several constant fields are added (see Figure T.3.16a). Please type in the seven fields as shown beneath.

Step name:

Fields:

#	Name	Type	Form	Leng	Preci	Currency	Decimal	Group	Value
1	SourceInstitutionID	String							
2	SourceID	String							
3	UnitID	String							
4	eseXml	String							
5	recordOK	Boolean							
6	filename	String							:::NO UNITS IN INPUT FILE:::
7	fileid	String							:::NO UNITS IN INPUT FILE:::

OK Cancel

Figure T.3.16a Constant fields added to the Transformation

Not the values for “filename” and “fileid”. Because no name or id can be created for a record without input the values are “:::NO UNITS IN INPUT FILE:::”.

When this is done click “OK” and go on with the Transformation.

**T.3.17 “Select values” – remove input file 2**

Figure T.3.17 shows the progress in our Transformation. As you can see another “Select values” step called “remove input file 2” has been created and connected with the last “Add constant” step. Its use is to remove the input file of the incorrect records without input.

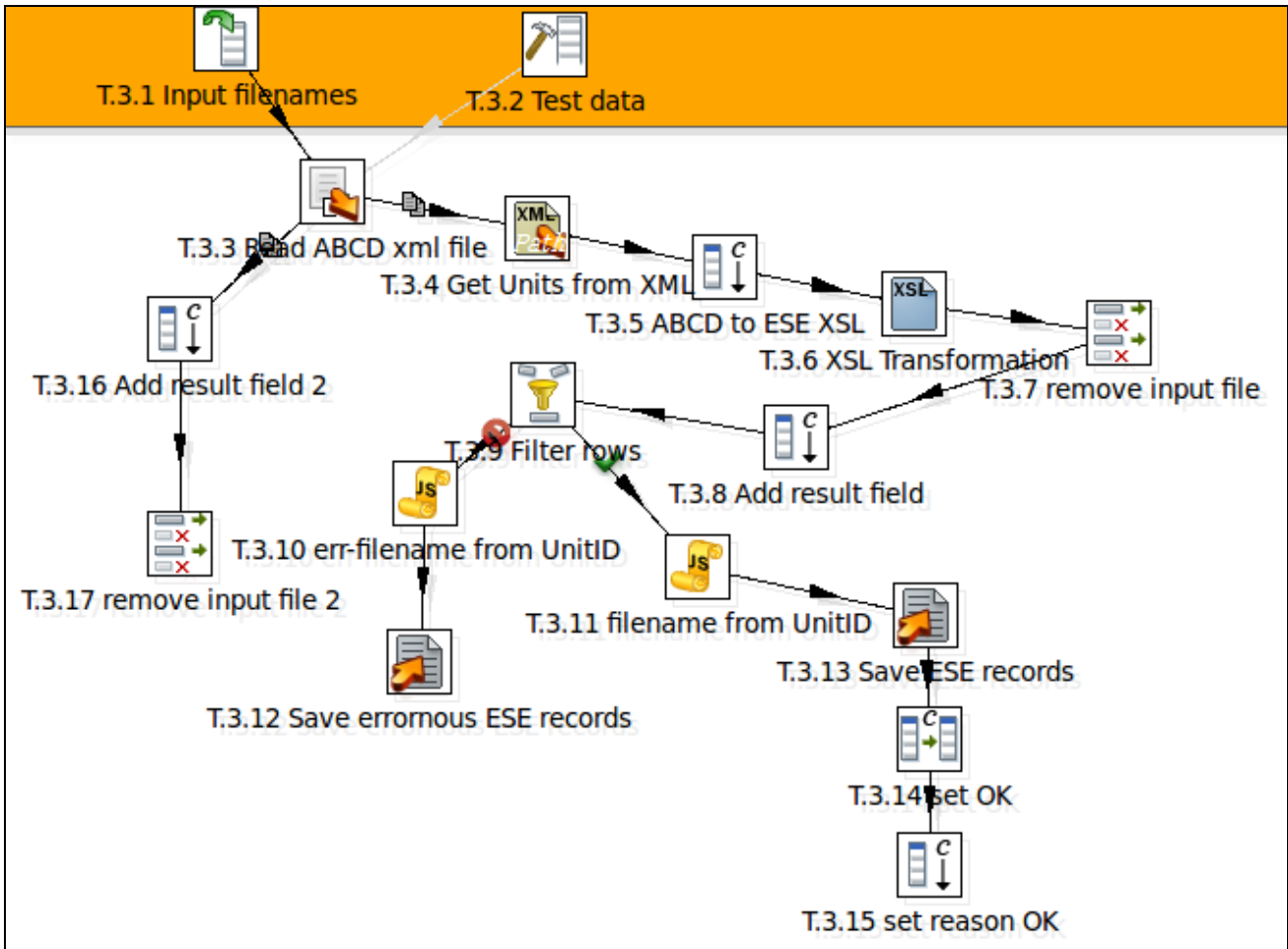


Figure T.3.17 Adding a "Select values" step to remove the input file

Figure T.3.17a shows that the field "abcdXML" has been selected under the Remove section. It contains the ABCD record in form of an XML document that we do not need anymore. When this is done click "OK".

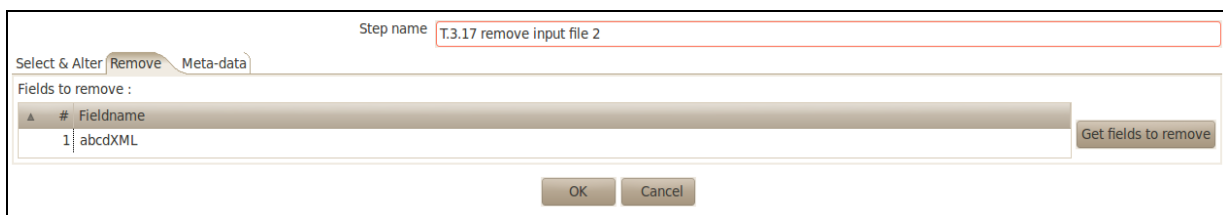


Figure T.3.17a Removing the second input file

**T.3.18 "set field value to a constant" – set NOT OK**

With this step it is possible – as the name reveals – to set the value of a field to a constant value. First the step has to be added to our Transformation to the error handling path. Note that two steps are connected to T.3.18 (Figure T.3.18).

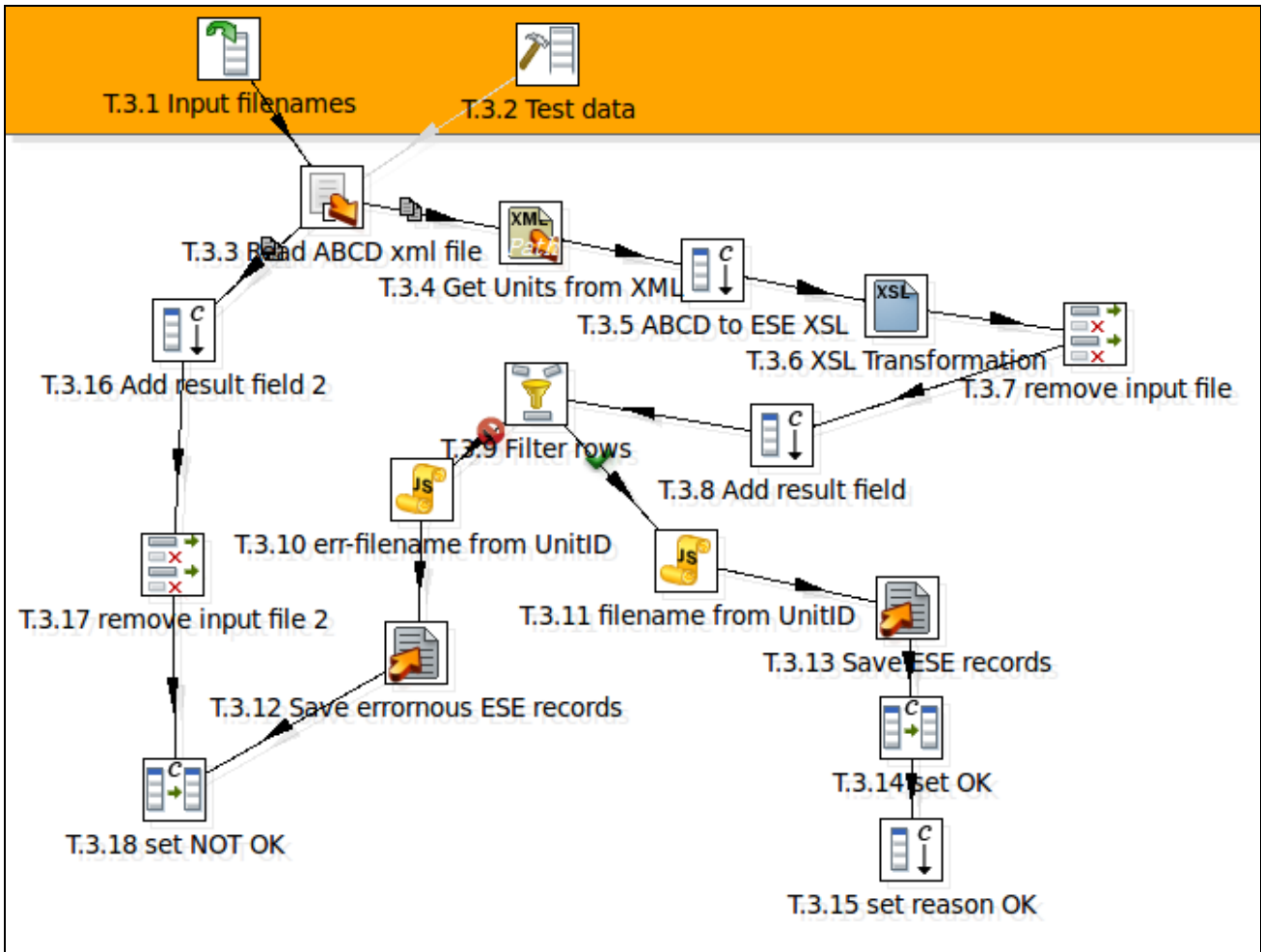


Figure T.3.18 Adding a "Set field value to constant" step

In Figure T.3.18a you can see that the field "recordOK" was chosen. This field is replaced by the value "false" to show that the record is NOT OK.

Step name

Use variable in constant

Fields

#	Field	Replace by value	Conversion mask (Date)
1	recordOK	false	

OK Get Fields Cancel

Figure T.3.18a "set NOT OK" with the value "false"

**T.3.19 Get data from XML – Get error Reason**

To find out the reason for the error files another “Get data from XML” step is added to T.3.18 (see Figure T.3.19).

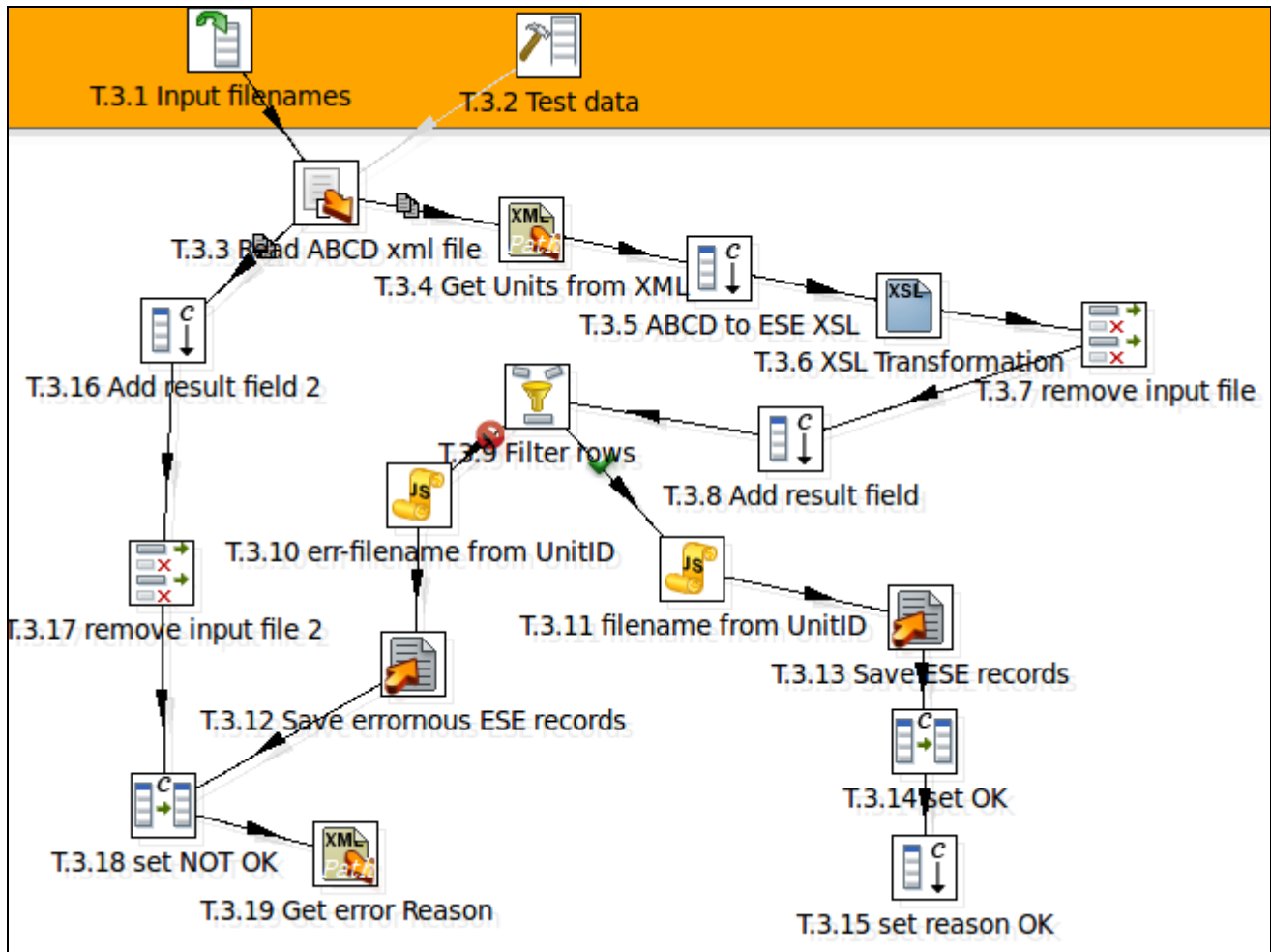


Figure T.3.19 Adding a “Get data from XML” step to find the error reason

Figure T.3.19a shows the File section of the step. Again the option “XML source is defined in a field?” has to be selected and the field “eseXML” has to be chosen after “get XML source from a field”.

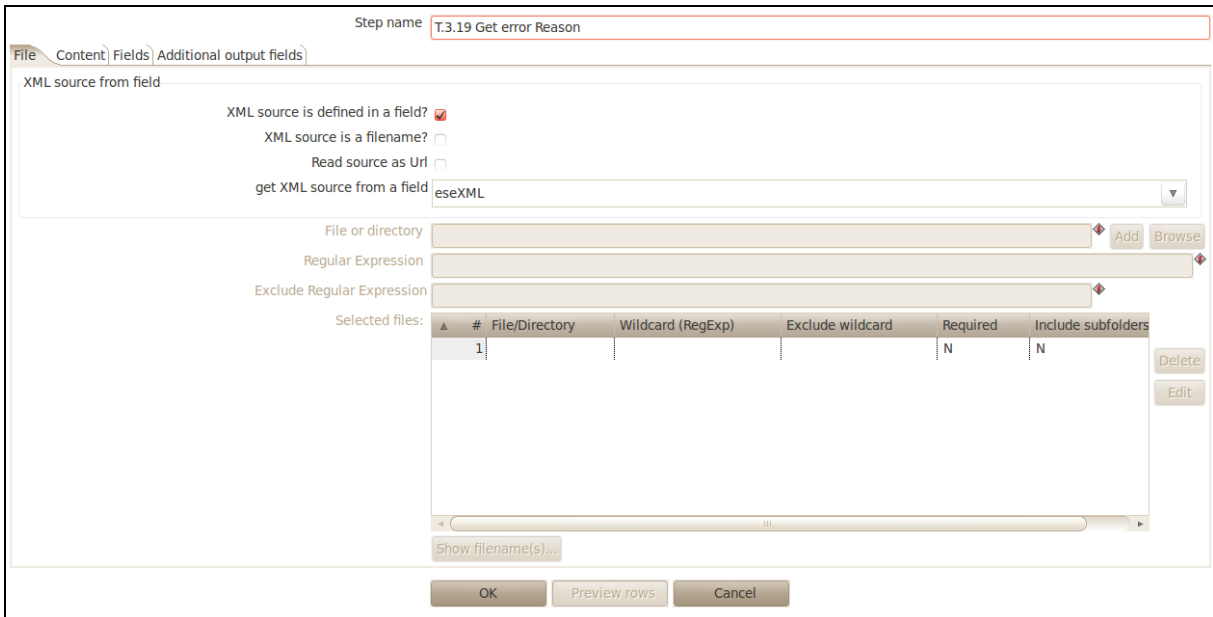


Figure T.3.19a File section of "Get error Reason"

In the Content section (see Figure T.3.19b) the XPath statement has to be written after "Loop XPath". In this step it is "/" which selects all nodes. Do not forget to check "Ignore empty file" and "Do not raise an error if no files".

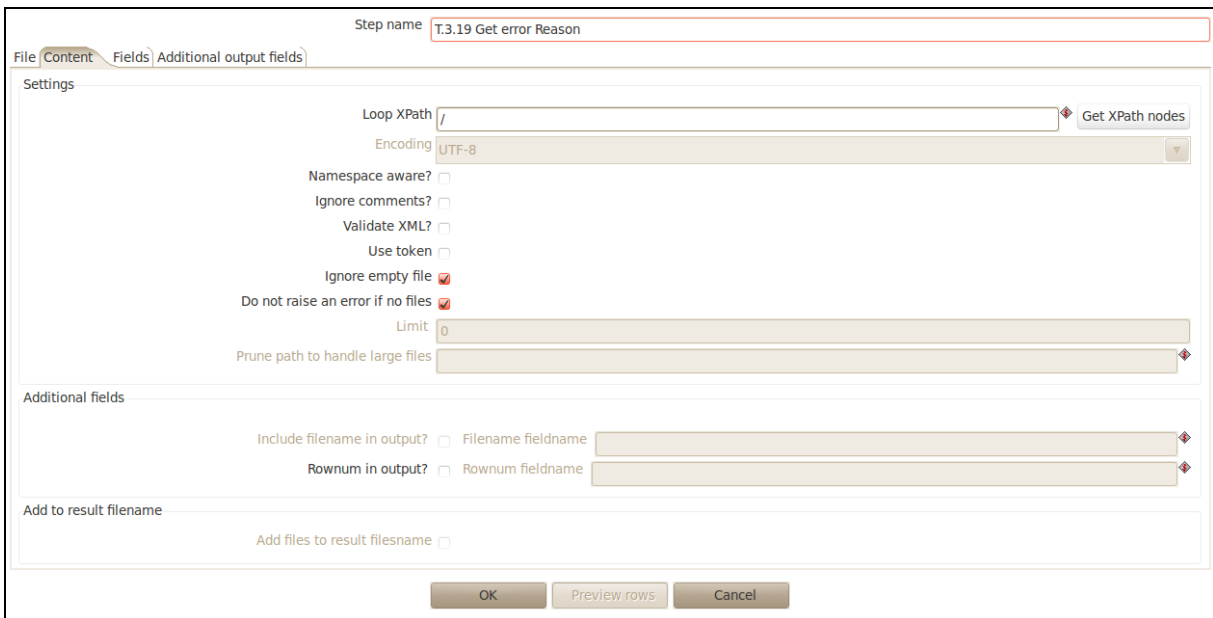


Figure T.3.19b Content section of "Get error Reason"

Now the Fields section is missing (see Figure T.3.19c). You can see the name of the field is "errorReason". The XPath defines the position of this value: "/europeana\_record@\_record\_invalid\_reason". It is important to choose "Attribute" in the column "Element".

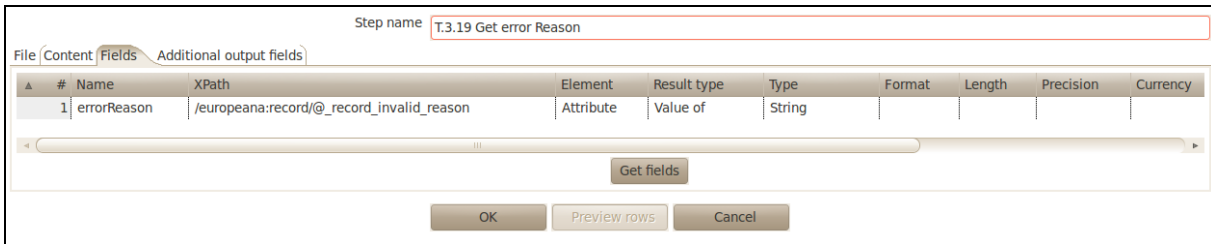


Figure T.3.19c Fields section of "Get error Reason" with field "errorReason"

When everything is entered correctly move on to the next step.

### T.3.20 Select values –select result

To select the fields needed for a result and to connect the "right" and the "wrong" path of the records another "Select values" step has to be added. As you can see in Figure T.3.23, steps T.3.19 and T.3.22 are connected with this step. Please take care to choose the right direction for the hops.

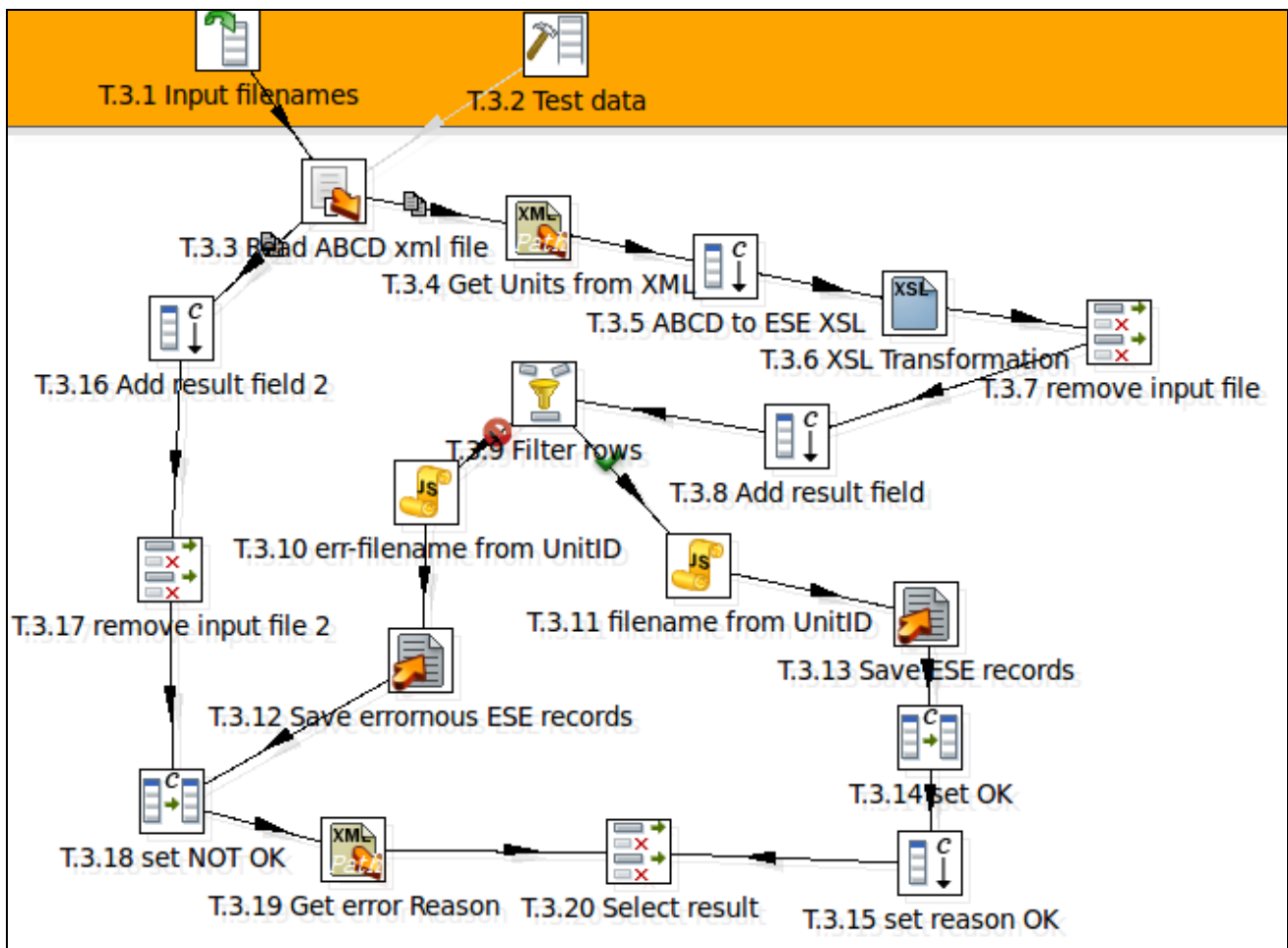


Figure T.3.20 Adding "Select result" and connecting the steps

Figure T.3.20a shows the fields that are chosen: "abcdFilename", "filename", "recordOK", "errorReason", "SourceInstitutionID", "SourceID", "UnitID" and "eseLogDir". When all fields are chosen click "OK".

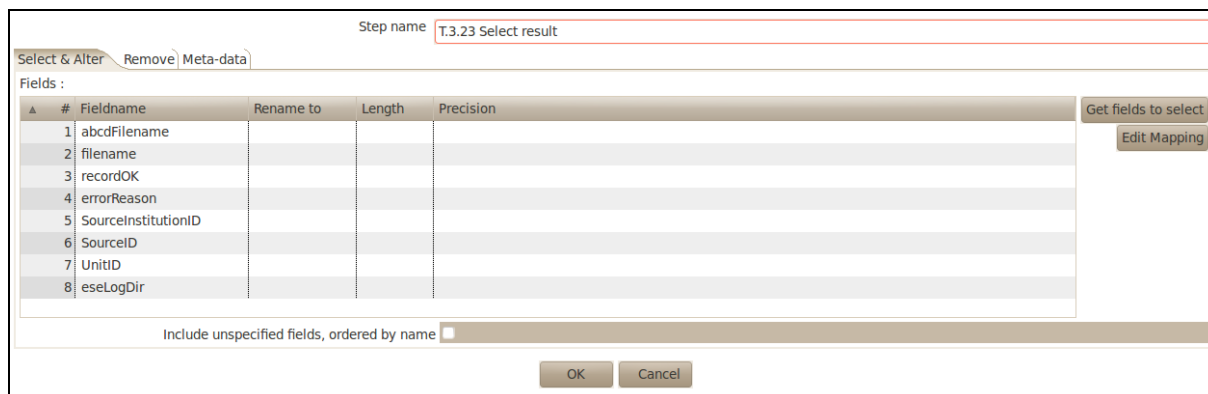


Figure T.3.20a Selected fields for result

### T.3.21 Modified Java Script value – logfile name with date

Now another “Modified Java Script value” step has to be added to create the name of the logfile with the date (see Figure T.3.21)

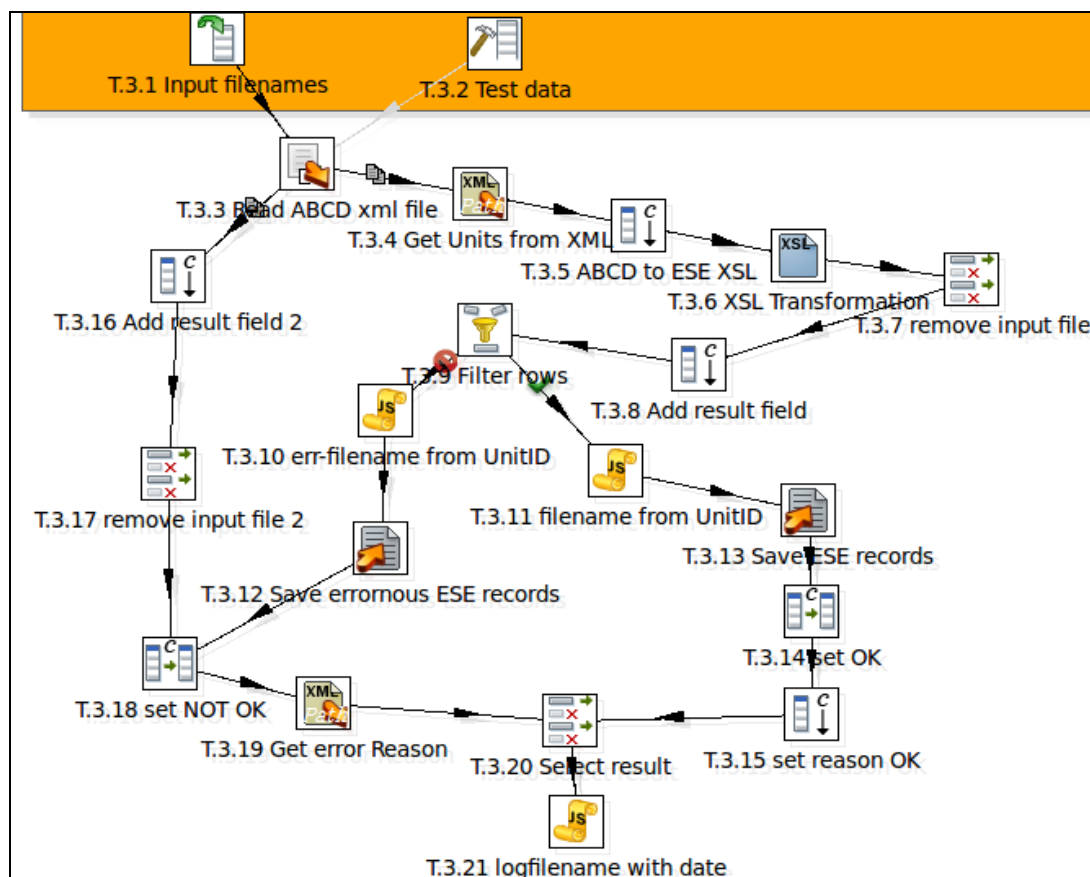


Figure T.3.21 Adding “Modified Java Script value” to create logfile name with date

As you already know from the other Java Script steps you have to define a script and fields. Figure T.3.21a shows the settings.

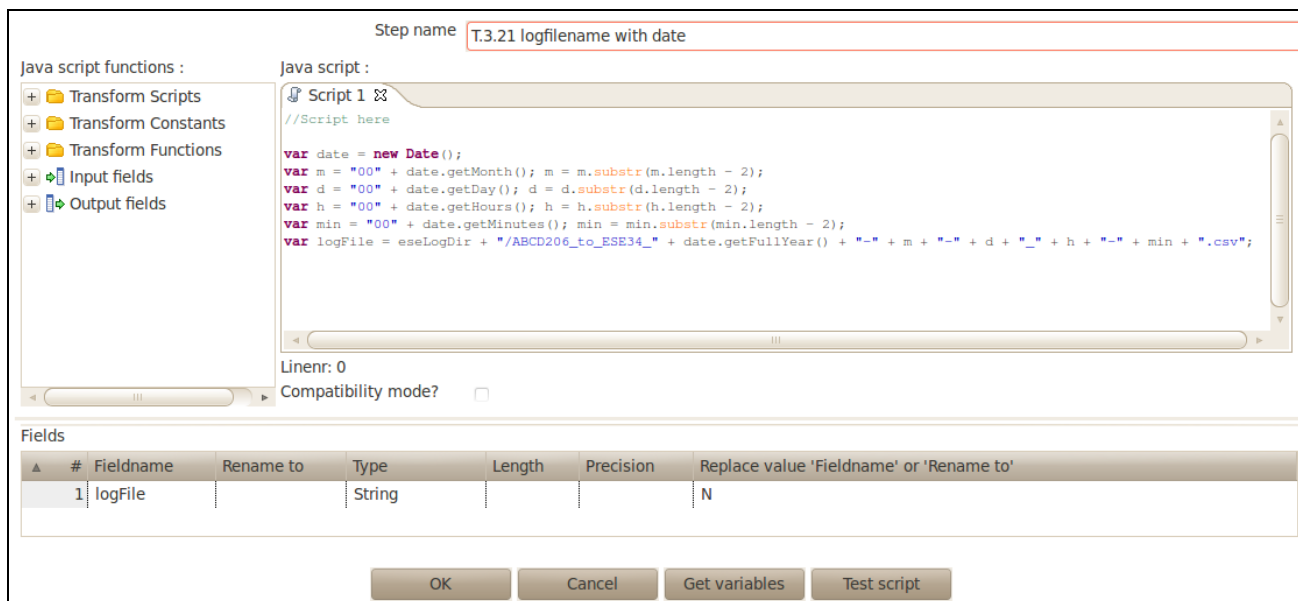


Figure T.3.21 a Creating logfilename with date with Java Script

The Script is:

```

var date = new Date();

var m = "00" + date.getMonth(); m = m.substr(m.length - 2);

var d = "00" + date.getDay(); d = d.substr(d.length - 2);

var h = "00" + date.getHours(); h = h.substr(h.length - 2);

var min = "00" + date.getMinutes(); min = min.substr(min.length - 2);

var logfile = eseLogDir + "/ABCD206_to_ESE34_" + date.getFullYear() + "-" + m + "-" + d + "_" + h + "-" + min + ".csv";
    
```

Beneath you have to type the fieldname "logfile" with its type "String". When this is done click "OK" and move on to the final part of the third Transformation.

**T.3.22 Text file output – Log: ABCD206\_to\_ESE34\_<date>.csv**

This step creates the logfiles we specified with Java Script the step before. As you can see in Figure T.3.22 a "Text file output" step is added. The name of the logfiles will be "ABCD206\_to\_ESE34\_<date>.csv".



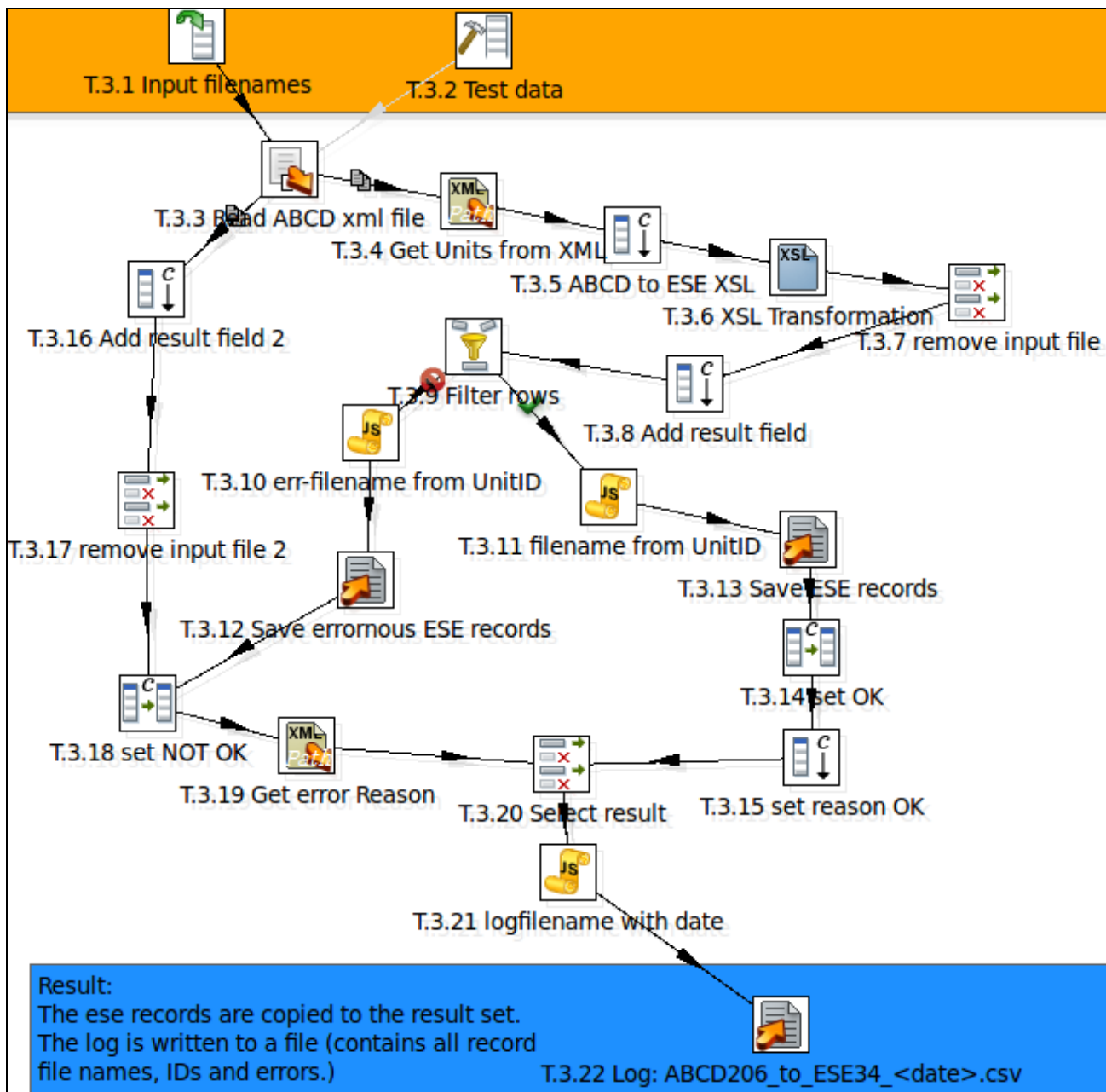


Figure T.3.22 Adding another "Text file output" step to create the logfiles

Figure T.3.22a shows the settings of the File section. Like in other text file steps before the option "Accept file name from field?" has to be activated and beneath the field "logfile" has been chosen after "File name field".

Then you can move on to the Content section (see Figure T.3.22b). As you can see "Append" and "Header" are checked. Moreover the "Enclosure" is defined with a quotation mark and "Format", "Compression" and "Encoding" have been selected.

Step name: T.3.22 Log: ABCD206\_to\_ESE34\_<date>.csv

File Content Fields

Filename: /tmp/ABCD206\_to\_ESE34\_result Browse...

Run this as command instead?

Pass output to servlet

Create Parent folder

Do not create file at start

Accept file name from field?

File name field: logFile

Extension:

Include stepnr in filename?

Include partition nr in filename?

Include date in filename?

Include time in filename?

Specify Date time format

Date time format: yyyy-MM-dd\_HH-mm

Show filename(s)...

Add filenames to result

OK Cancel

Figure T.3.22a File section of "Log: ABCD206\_to\_ESE34\_<date>.csv"

Step name: T.3.22 Log: ABCD206\_to\_ESE34\_<date>.csv

File Content Fields

Append

Separator:  Insert TAB

Enclosure: "

Force the enclosure around fields?

Disable the enclosure fix?

Header

Footer

Format: LF terminated (Unix)

Compression: None

Encoding: UTF-8

Right pad fields

Fast data dump (no formatting)

Split every ... rows: 0

Add Ending line of file:

OK Cancel

Figure T.3.22b Content section of the "Text file output" step that creates a logfile

Now only the Fields section is missing (see Figure T.3.22c). The following fields are needed for the logfiles: "SourceID", "SourceInstitutionID", "UnitID", "recordOK", "errorReason", "filename" and "abcdfilename".

When this is done click "OK". Now only one step is missing and the third Transformation is finished.

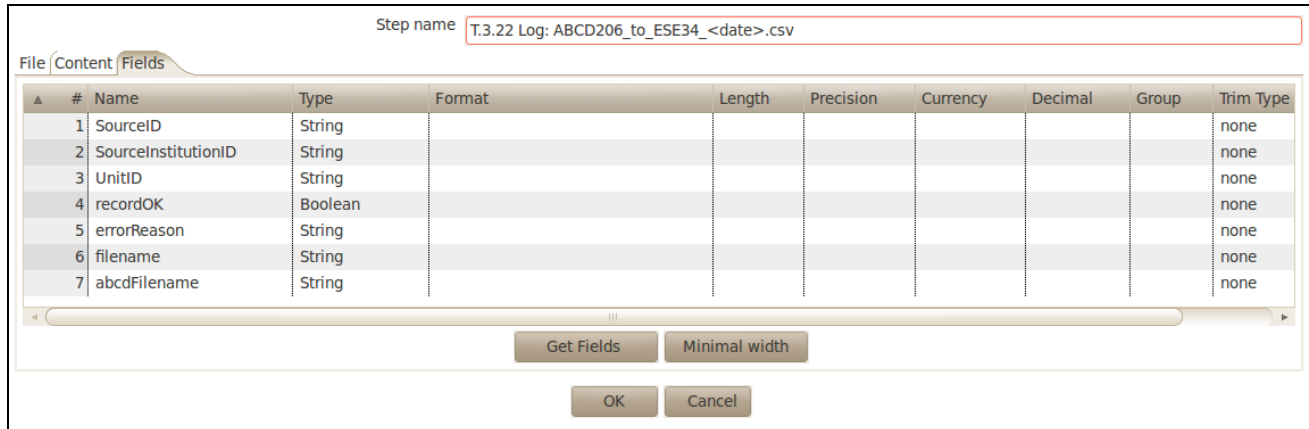


Figure T.3.22c Fields section of "Log: ABCD206\_to\_ESE34\_<data>.csv"

### T.3.23 Copy rows to result

As done before in the other Transformation the results created in the Transformation need to be transferred to the Job. To achieve this a "Copy rows to result" step is added (see Figure T.3.26) and connected with T.3.23. When this is done the Transformation is ready and should look like in Figure T.3.

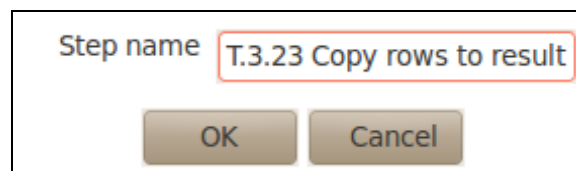


Figure T.3.23 Copy rows to result

### J.1 The Pentaho Job "Biocase\_Harvest\_to\_ESE" (II)

Figure J.4 gives again an overview of the Job. Now that we have created the three Transformations we have to connect them in a Job. To achieve this we need to perform the following steps.

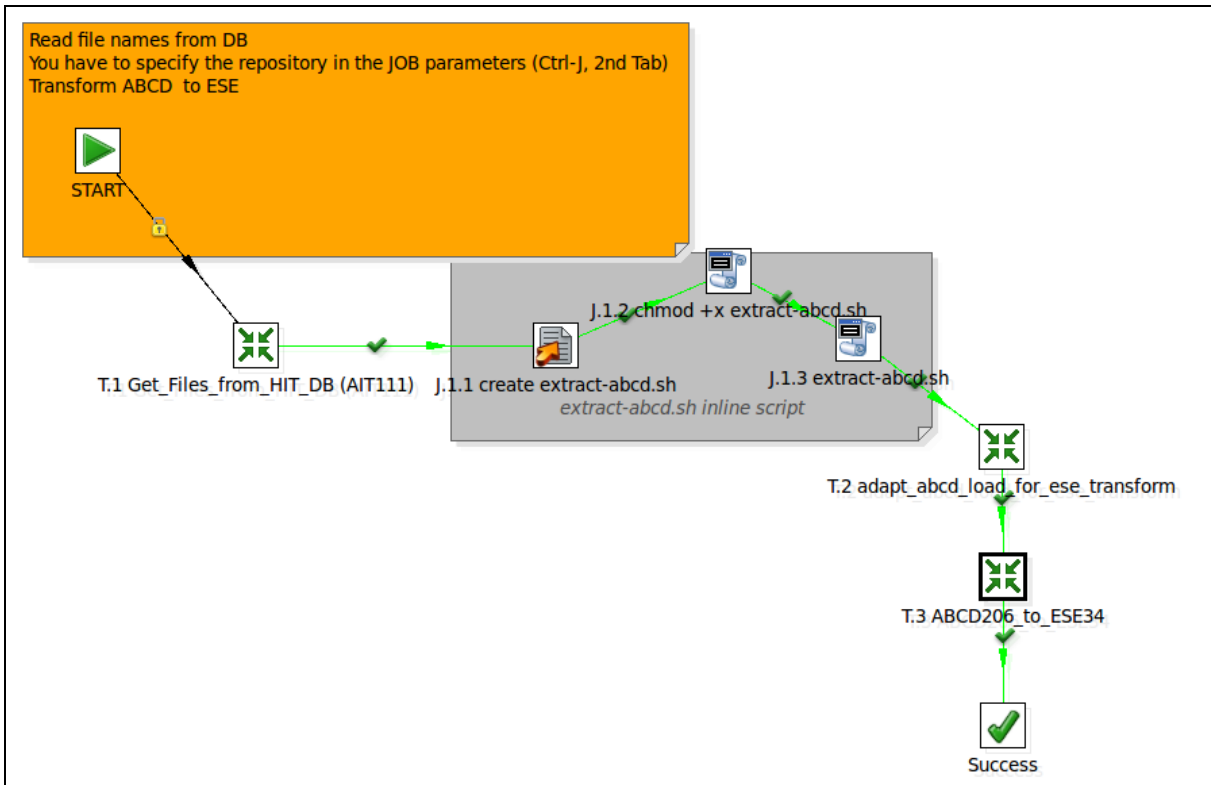


Figure J.4 The Pentaho Job (II)

### J.1.T.1 Get\_Files\_from\_HIT\_DB (AIT111) and START

Every Job in Pentaho begins with the START icon. This step will be connected to our first transformation called “Get\_Files\_from\_HIT\_DB (AIT111)” (Figure J.1a).

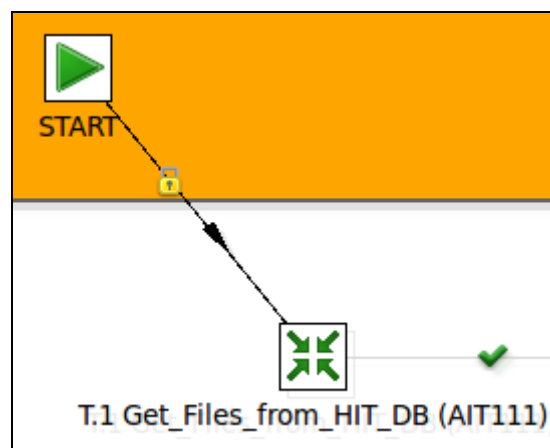


Figure J.1a First two steps in the Pentaho Job

When double-clicking on the START icon you can determine whether you want to schedule your Job (Figure J.1a). In our example there is no scheduling.

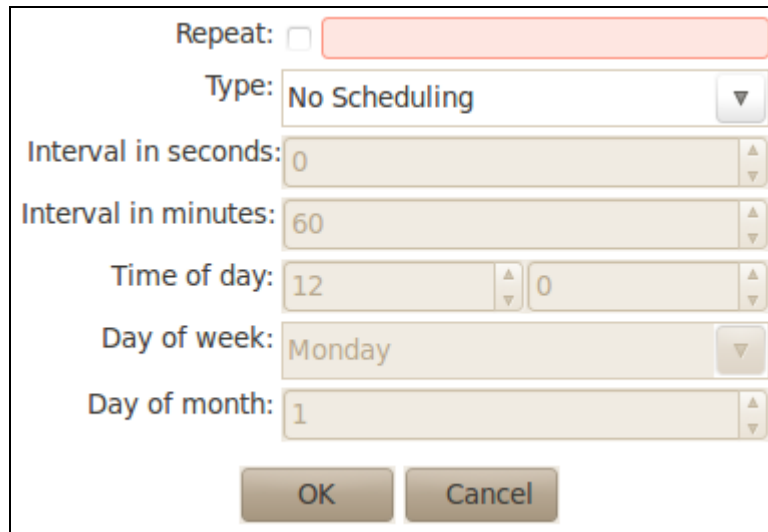


Figure J.1b Scheduling a Job

Like the START step you can find the step “Transformation”<sup>28</sup> in the “General” directory. With this step it is possible to integrate our first Transformation we have created before. With a double-click the menu opens (Figure J.1.T.1a).

Select “Specify by reference” and choose your first Transformation from the repository. After that you can move on to the “Parameters” tab (Figure J.1.T.1b). Make sure you select “Pass all parameter values down to the sub-transformation”.

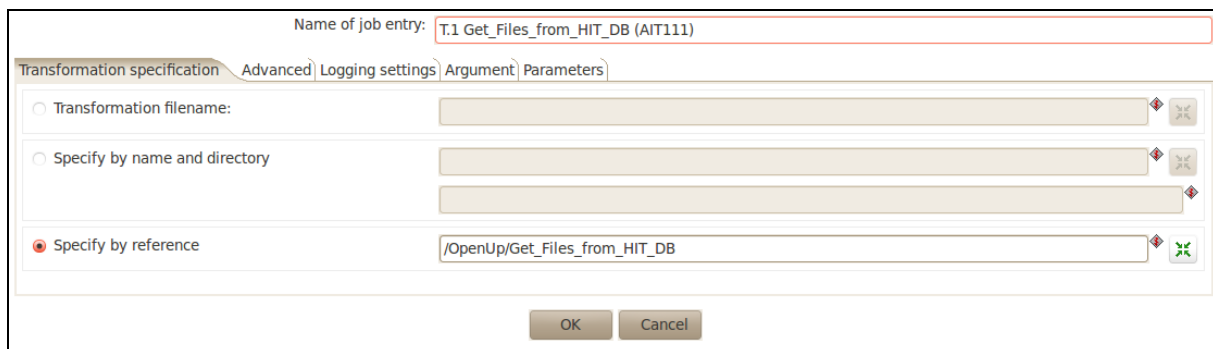


Figure J.1.T.1a Transformation specification

When selecting “Get parameters” the three parameters “base\_dir”, “name” and “uddi\_key” appear. Now you have to define the value for each by typing them in the column “Value”.

<sup>28</sup> <http://wiki.pentaho.com/display/EAI/Transformation+%28job+entry%29>

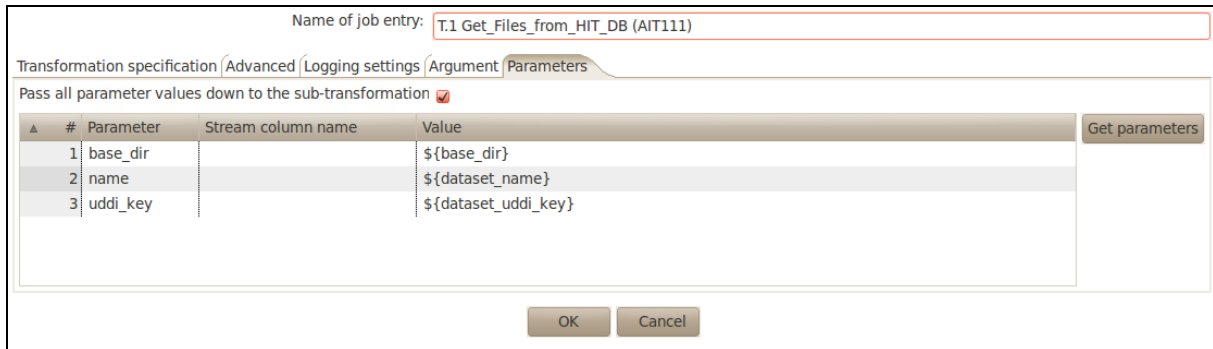


Figure J.1.T.1b Parameters of T.1

When this is done click “OK” and move on to the next step.

### J.1.1 Write to file – create extract-abcd.sh

The next step that needs to be added to the Job is called “Write to file” which was renamed in our example to “create extract-abcd.sh”. It can be found under the “File Management” directory. Figure J.1.1 shows the progress in the Job.

With this step you can write text content to a file. We use it – together with the next two steps - to create a shell script<sup>29</sup> that extracts ABC Data. After double-clicking on the icon, you can see the text in our example (Figure J.1.1a). Your file path in the text with the .tmp file will differ from the one in our example. You also have to choose a File name and the Encoding.

For choosing the filename it is very important to select the directory of the Pentaho software (Pentaho/data integration). “/tmp/extract-abcd.sh” shows the continuing path with the name of our future shell script: “extract-abcd.sh”.

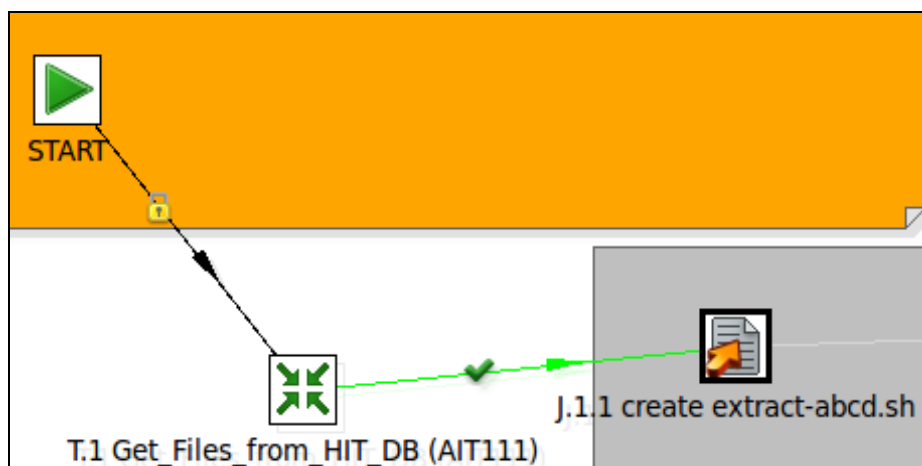


Figure J.1.1 Adding the “Write to file” step to the job

<sup>29</sup> <http://www.freeos.com/guides/lsst/>



Figure J.1.1a Settings of the “Write to file” step

When this is done click “OK” and go on with the shell script.

### J.1.2 Shell – chmod + x extract-abcd.sh

The next step you have to add to your Job is a “Shell” step (Figure J.1.2) that executes a shell script. You can find it under the “Scripting” directory. In our example the step is called “chmod + x extract-abcd.sh”.

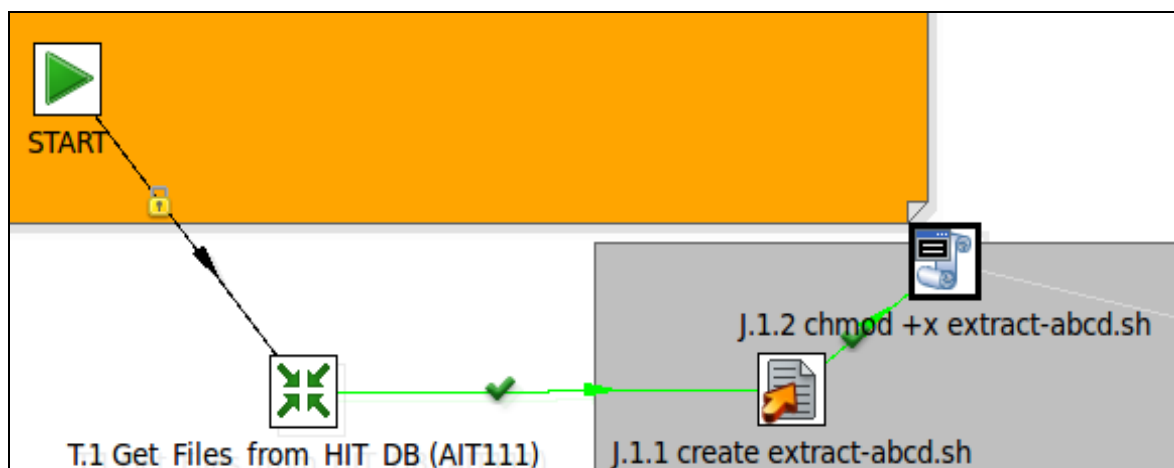
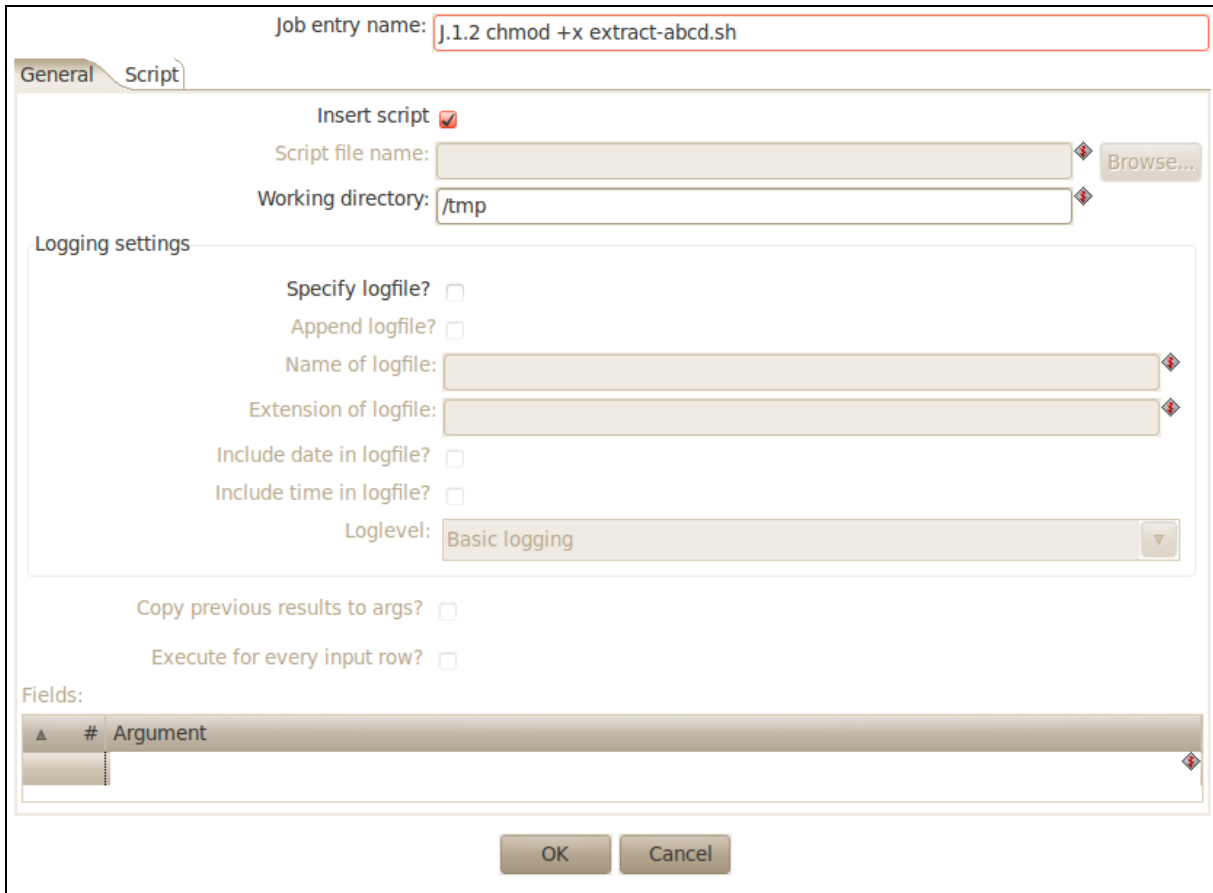


Figure J.1.2 Adding a “Shell” step to the job

After double-clicking on the icon, you will see the following window (Figure J.1.2a).



Job entry name: J.1.2 chmod +x extract-abcd.sh

General Script

Insert script

Script file name:  Browse...

Working directory: /tmp

Logging settings

Specify logfile?

Append logfile?

Name of logfile:

Extension of logfile:

Include date in logfile?

Include time in logfile?

Loglevel: Basic logging

Copy previous results to args?

Execute for every input row?

Fields:

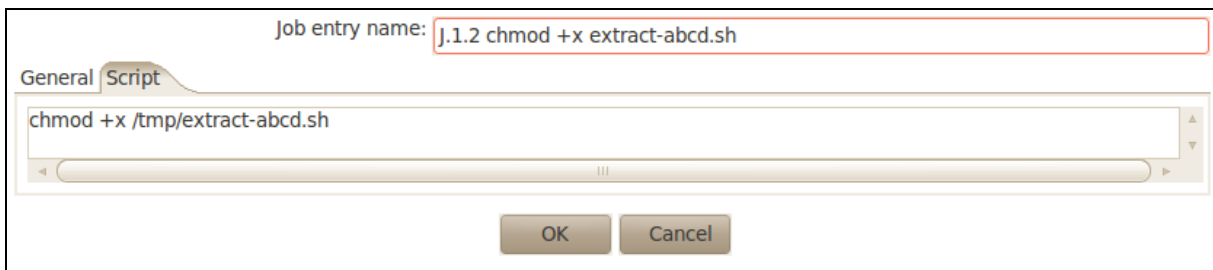
#	Argument

OK Cancel

Figure J.1.2a General section of the step

As you can see this step has two sections: "General" and "Script". In the General section (Figure J.1.2a) the "Insert Script" option has to be activated because we will insert a script afterwards and the "Working directory" has to be specified. In our example it is "/tmp" – the directory that contains our shell script "extract-abcd.sh".

After that move on to the "Script" tab (Figure J.1.2b).



Job entry name: J.1.2 chmod +x extract-abcd.sh

General Script

chmod +x /tmp/extract-abcd.sh

OK Cancel

Figure J.1.2b Script of "Execute a shell script"

It says "chmod + x /tmp/extract-abcd.sh"

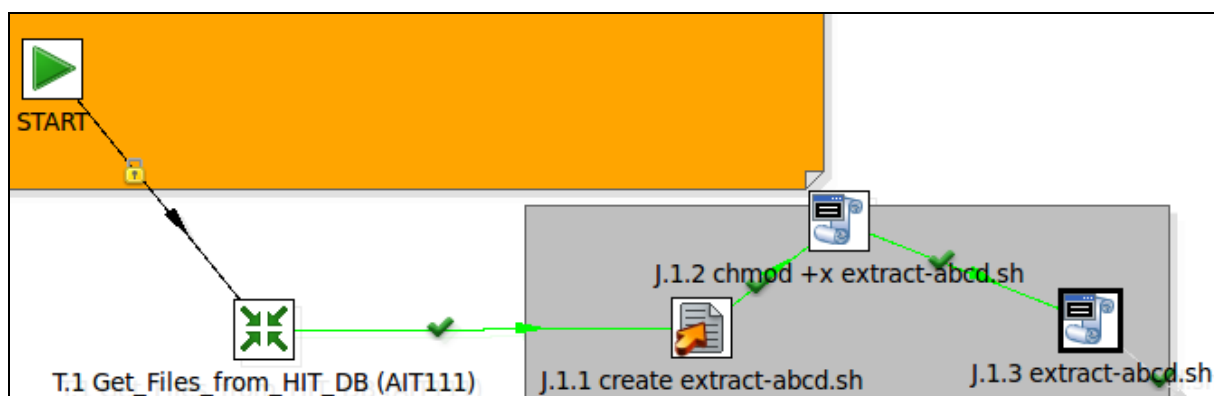


“chmod” is a Unix command that defines the access permitted to a file<sup>30</sup>. the “+ x” makes it possible to execute a file.

After clicking “OK” we need another “Shell” step.

### J.1.3 Shell – extract-abcd.sh

The Job with the second shell script step can be seen in *Figure J.1.3*. In our example its name is “extract-abcd.sh”.

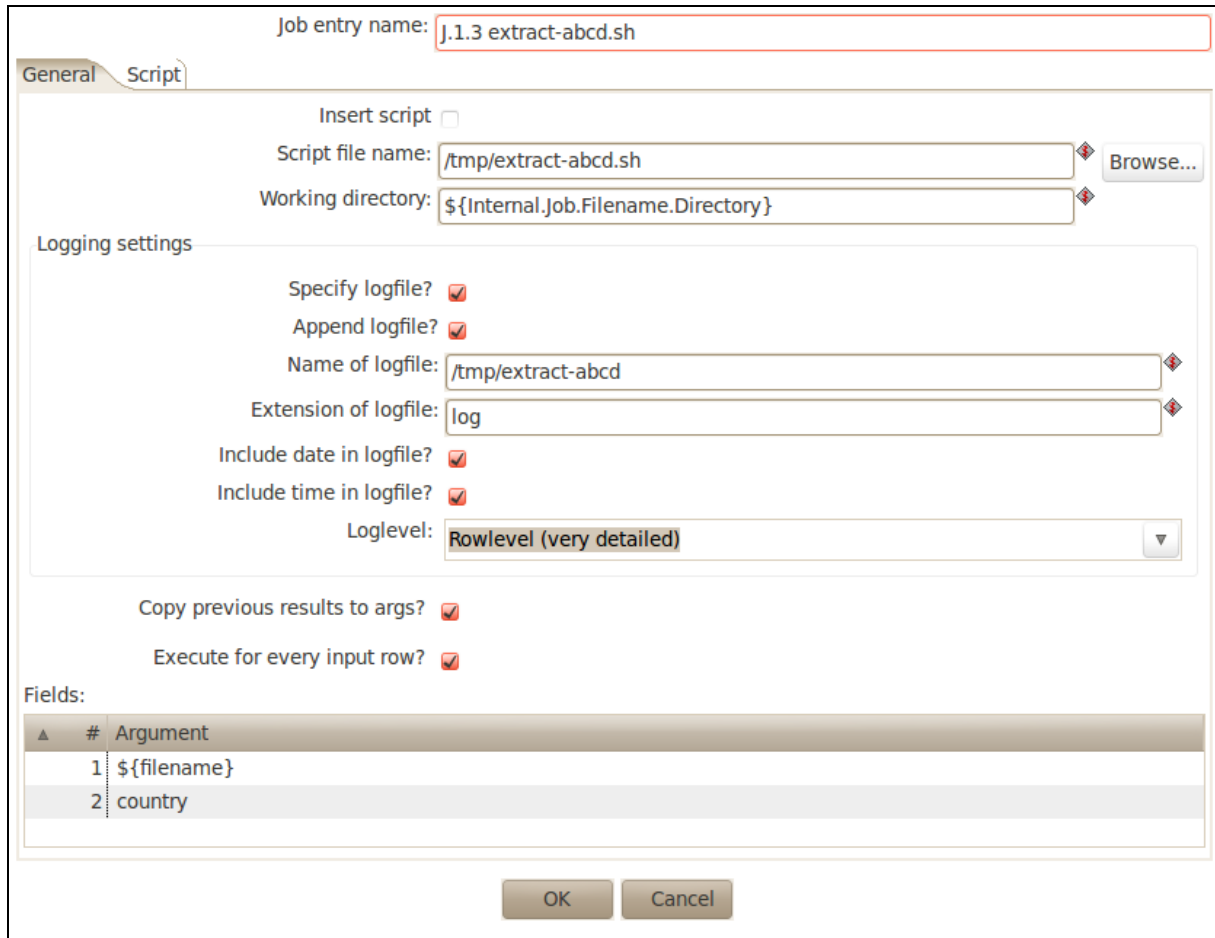


*Figure J.1.3 Add another “Shell” step to our job*

Figure *J.1.3a* shows the General section for this step. As you can see the “Script file name” is the one from the last step: /tmp/extract-abcd.sh. The “Working directory” is defined with the variable `${Internal.Job.Filename.Directory}`. A list of variables appear if you press CTRL + SPACE while positioning the cursor on the gray and red symbol beside the line.

Then you can go on to the “Logging settings” beneath. As you can see we want to “Specify logfile?” and “Append logfile?”. Now we need the “Name of logfile” which is “/tmp/extract-abcd” and the “Extension of logfile” which is “log”. Moreover the options “Include date in logfile” and “Include time in logfile” are activated and the “Loglevel” is “Rowlevel (very detailed)” like before.

<sup>30</sup> <http://en.wikipedia.org/wiki/Chmod>



Job entry name: J.1.3 extract-abcd.sh

General Script

Insert script

Script file name: /tmp/extract-abcd.sh Browse...

Working directory: \${Internal.Job.Filename.Directory}

Logging settings

Specify logfile?

Append logfile?

Name of logfile: /tmp/extract-abcd

Extension of logfile: log

Include date in logfile?

Include time in logfile?

Loglevel: Rowlevel (very detailed)

Copy previous results to args?

Execute for every input row?

Fields:

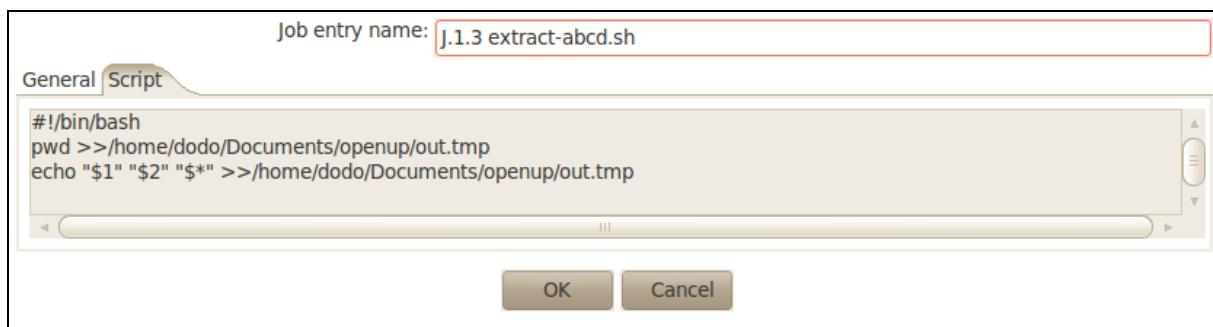
#	Argument
1	\${filename}
2	country

OK Cancel

Figure J.1.3a General section for “extract-abcd.sh”

Furthermore you have to choose “Copy previous results to args?” and “Execute for every input row?”. Before you can click “OK” you have to fill in the Fields section with the Arguments `${filename}` and `country`.

Before you can move on to the Script tab you have to select “Insert Script” on top of the window. When this is done click on the Script tab (Figure J.1.3b).and fill in the Script.



Job entry name: J.1.3 extract-abcd.sh

General Script

```
#!/bin/bash
pwd >>/home/dodo/Documents/openup/out.tmp
echo "$1" "$2" "$*" >>/home/dodo/Documents/openup/out.tmp
```

OK Cancel

Figure J.1.3b Script of “extract-abcd.sh”

Like in the previous step the file path in your example will differ from the one shown in *Figure J.1.3b*. When this is done go back to the General section and deactivate the “Insert script” option.

After that click “OK” and move on to the next step.

### J.1.T.2 Transformation – adapt\_abcd\_load\_for\_ese\_transform

Now it is time to fill in the second Transformation “adapt\_abcd\_load\_for\_ese\_transform” (*Figure J.1.T.2*).

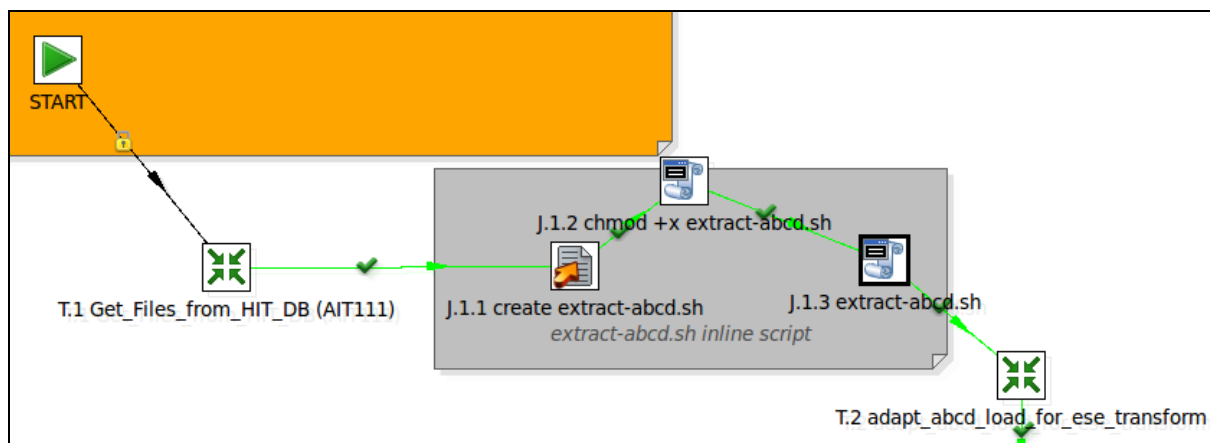


Figure J.1.T.2 Adding the “adapt\_abcd\_load\_for\_ese\_transform”

We do this by dragging another “Transformation” step on the canvas and connect it with the last step. *Figure J.1.T.2a* shows how the right Transformation is chosen.

Figure J.1.T.2a Transformation specification for “adapt\_abcd\_load\_for\_ese\_transform”

Like before you have to choose “Specify by reference” and the transformation in your repository by clicking on the green symbol on the right. After that you can move on to the “Advanced” register (see *Figure J.1.T.2b*).

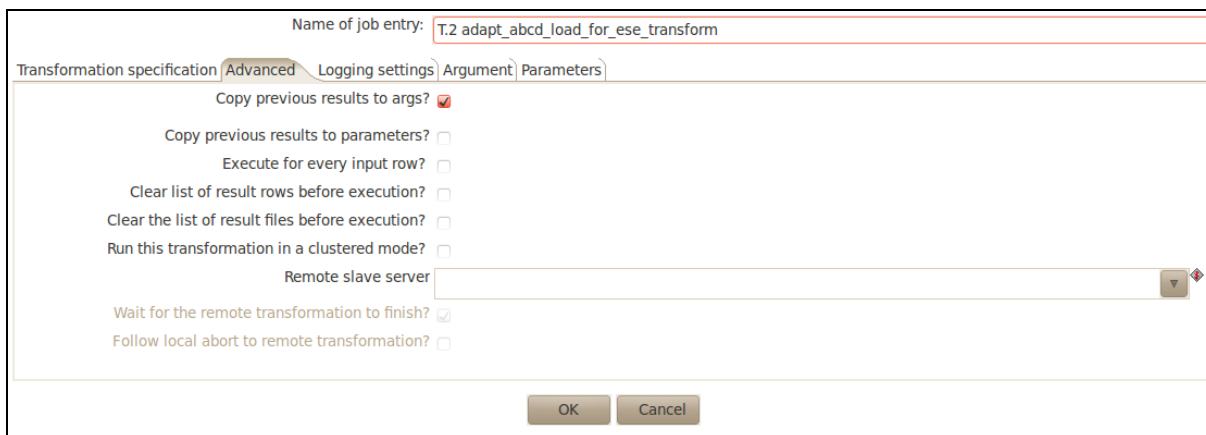
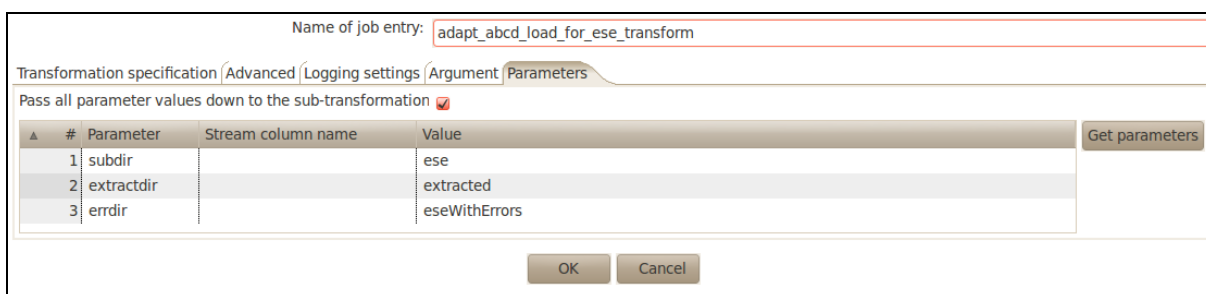


Figure J.1.T.2b Advanced settings for “adapt\_abcd\_load\_for\_ese\_transform”

Here the only thing you have to do is to activate the option “Copy previous results to args?”. After this is done move on to “Parameters” (see Figure J.1.T.2c).



#	Parameter	Stream column name	Value
1	subdir		ese
2	extractdir		extracted
3	errdir		eseWithErrors

Figure J.1.T.2c Parameters of “adapt\_abcd\_load\_for\_ese\_transform”

In this case our parameters are “subdir” with the value “ese” and “extractdir” with the value “extracted”. When clicking on “Get parameters” the two fields appear. The values have to be chosen by yourself.

Please note you have to tick the option “Pass all parameter values down to the sub-transformation” above. We do not need the “Argument” section here.

Now you can click on “OK” and the step for adding the second Transformation is finished.

### J.1.T.3 Transformation – ABCD206\_to\_ESE34

Now only the third Transformation is missing. Again you can add it with the “Transformation” step (see Figure J.1.T.3).

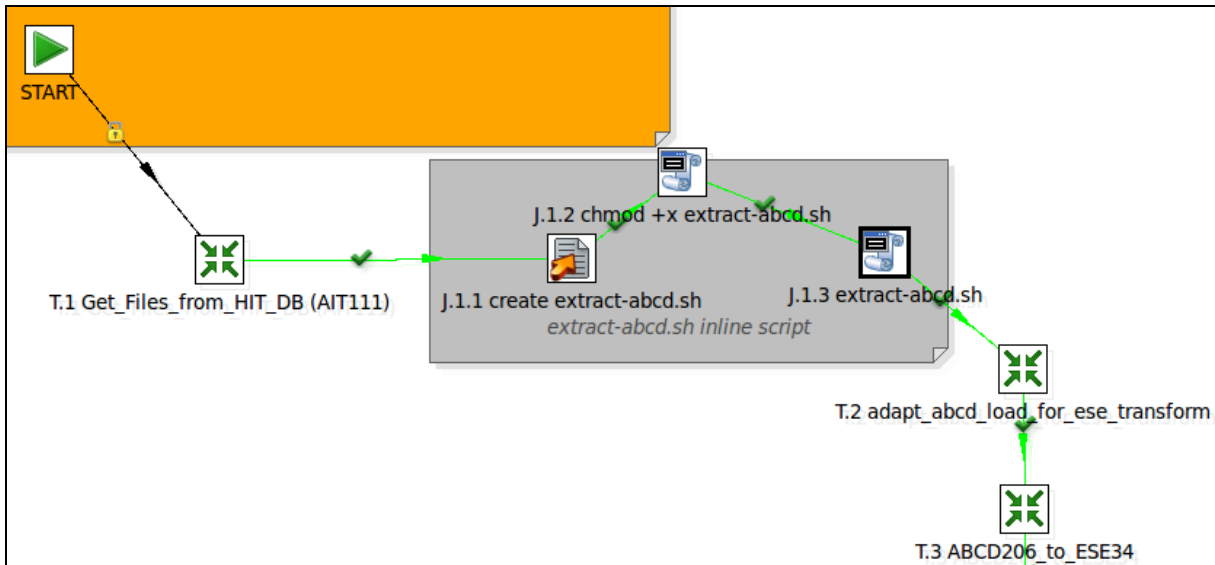


Figure J.1.T.3 Adding the “ABCD206\_to\_ESE34” transformation

Selecting this Transformation works like in the previous step (see *Figure J.1.T.3a*) with “Specify by reference and selecting the Transformation from the repository. Besides the “Transformation specification we need the tabs “Advanced” and the “Parameters”.

Figure J.1.T.3a Transformation specification

Figure J.1.T.3b shows the settings for Advanced. Here you have to select “Execute for every input row?”.

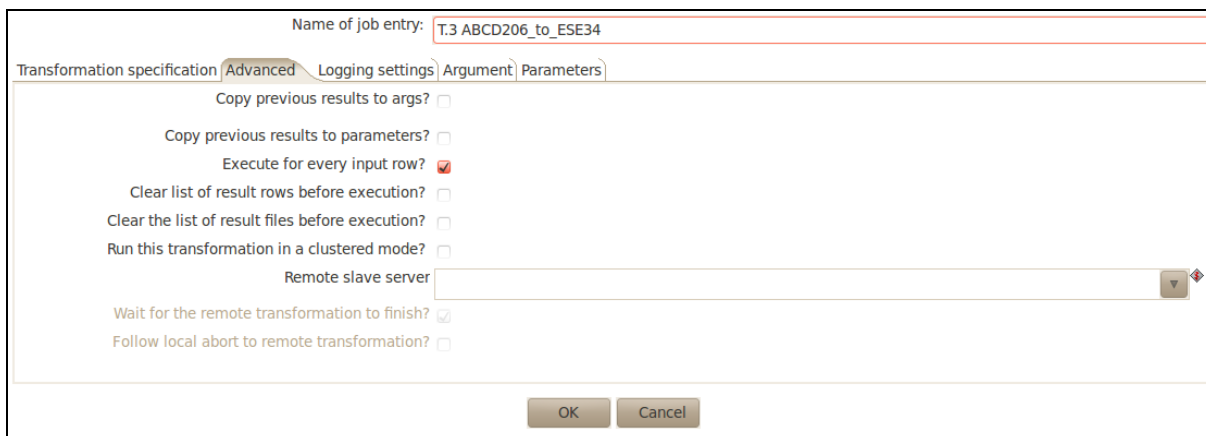
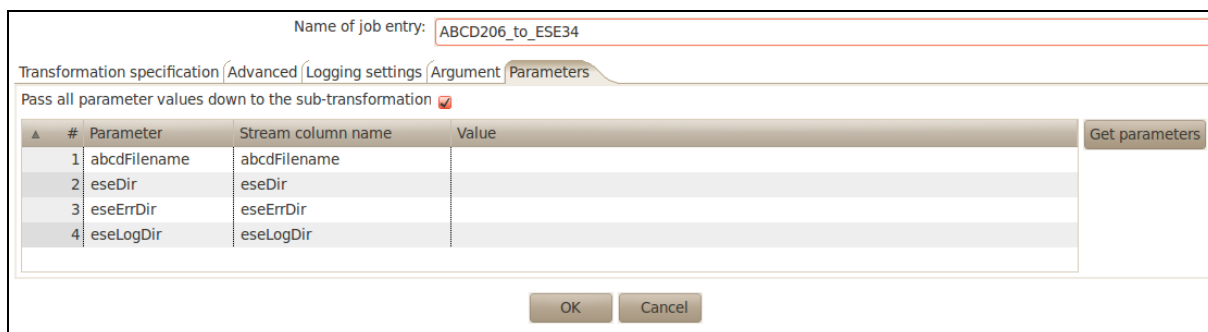


Figure J.1.T.3b Transformation specification for “ABCD206\_to\_ESE34”

The Parameters are shown in *Figure J.1.T.3c*. Like in the last Transformation you have to select “Pass all parameter values down to the sub-transformation”. Here the four parameters are “abcdFilename”, “eseDir”, “eseErrDir” and “eseLogDir”. The Stream column names besides are the same.



#	Parameter	Stream column name	Value
1	abcdFilename	abcdFilename	
2	eseDir	eseDir	
3	eseErrDir	eseErrDir	
4	eseLogDir	eseLogDir	

Figure J.1.T.3c Advanced section of “ABCD206\_to\_ESE34” transformation

When finished you can click “OK”.

Now you just have to add a “Success” icon (see *Figure J.1b*) and your job looks like the one shown in *J.1*.

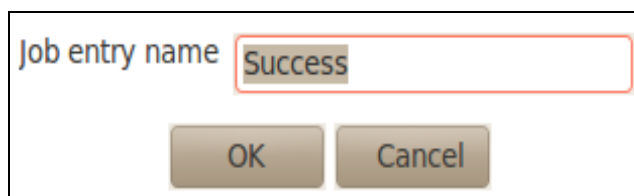


Figure J.1c Success

## 4. Pentaho example

In this chapter a specific example will run through the before described Transformations and the Job. So you will get a closer look at the input data (in ABCD schema) and the final output (an ESE record for every unit). Furthermore some preview results during the Pentaho process will be given.

### 4.1. The ABCD example record

In Figure 1–1 the schema of ABCD was shown in a very short form. In Figure 4–1 you can see one section of the ABCD file called Unit. One ABCD file contains many Units and for each unit the Pentaho process will create an ESE record. Remember the step “Get data from XML” where you had to specify in XPath which part of the XML document we needed. The XPath statement selected exactly the Units of each ABCD file.

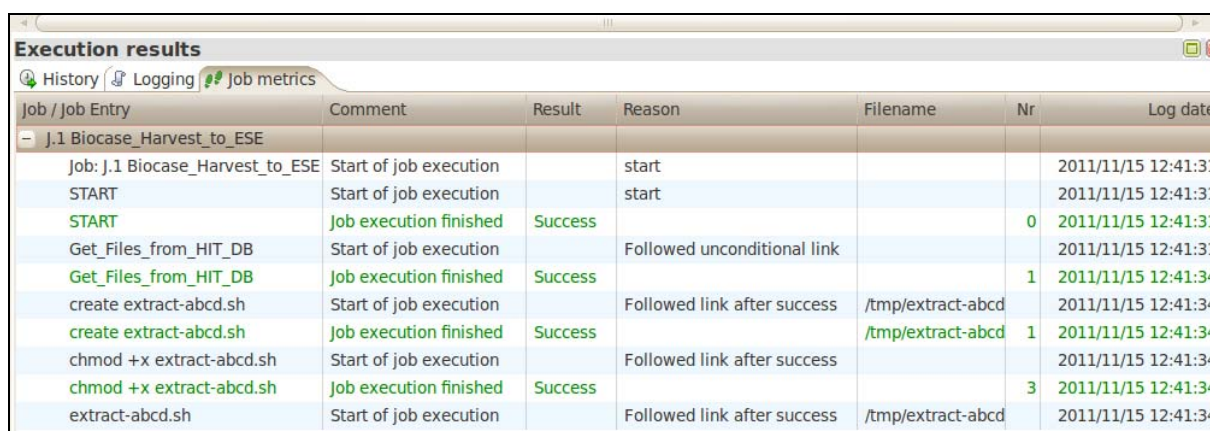
```
<abcd:Unit>
  <abcd:SourceInstitutionID>BGBM</abcd:SourceInstitutionID>
  <abcd:SourceID>Herbarium Berolinense</abcd:SourceID>
  <abcd:UnitID>B -W 13094 -00 0</abcd:UnitID>
  <abcd:Identifications>
    <abcd:Identification>
      <abcd:Result>
        <abcd:TaxonIdentified>
          <abcd:HigherTaxa>
            <abcd:HigherTaxon>
              <abcd:HigherTaxonName>LEGUMINOSAE</abcd:HigherTaxonName>
              <abcd:HigherTaxonRank>familia</abcd:HigherTaxonRank>
            </abcd:HigherTaxon>
          </abcd:HigherTaxa>
          <abcd:ScientificName>
            <abcd:FullScientificNameString>Abrus praecatorius</abcd:FullScientificNameString>
            <abcd:NameAtomised>
              <abcd:Botanical>
                <abcd:GenusOrMonomial>Abrus</abcd:GenusOrMonomial>
                <abcd:FirstEpithet>praecatorius</abcd:FirstEpithet>
              </abcd:Botanical>
            </abcd:NameAtomised>
          </abcd:ScientificName>
        </abcd:TaxonIdentified>
      </abcd:Result>
      <abcd:PreferredFlag>1</abcd:PreferredFlag>
    </abcd:Identification>
  </abcd:Identifications>
  <abcd:RecordBasis>PreservedSpecimen</abcd:RecordBasis>
  <abcd:KindOfUnit>herbarium sheet</abcd:KindOfUnit>
  <abcd:MultiMediaObjects>
    <abcd:MultiMediaObject>
      <abcd:FileURI>http://ww2.bgbm.org/herbarium/view_biocase.cfm?SpecimenPK=128757</abcd:FileURI>
      <abcd:Format>text/html</abcd:Format>
    </abcd:MultiMediaObject>
  </abcd:MultiMediaObjects>
  <abcd:Gathering>
    <abcd:NamedAreas>
      <abcd:NamedArea>
        <abcd:AreaClass language="en">Continent</abcd:AreaClass>
        <abcd:AreaName language="en" />
      </abcd:NamedArea>
    </abcd:NamedAreas>
  </abcd:Gathering>
</abcd:Unit>
```

```
<abcd:Notes language="en" />
</abcd:Gathering>
</abcd:Unit>
```

Figure 4–1 One Unit of an ABCD record

#### 4.2. The results of J.1 “Biocase\_Harvest\_to\_ESE”

If the Job has been created correctly you can see green Execution results after executing the Job (see Figure 4–2). This result contains the name of the Job and its steps, comments, result (“Success”), reason, filename, Nr. and a log date.



Job / Job Entry	Comment	Result	Reason	Filename	Nr	Log date
J.1 Biocase_Harvest_to_ESE						
Job: J.1 Biocase_Harvest_to_ESE	Start of job execution		start			2011/11/15 12:41:31
START	Start of job execution		start			2011/11/15 12:41:31
START	Job execution finished	Success			0	2011/11/15 12:41:31
Get_Files_from_HIT_DB	Start of job execution		Followed unconditional link			2011/11/15 12:41:31
Get_Files_from_HIT_DB	Job execution finished	Success			1	2011/11/15 12:41:34
create extract-abcd.sh	Start of job execution		Followed link after success	/tmp/extract-abcd		2011/11/15 12:41:34
create extract-abcd.sh	Job execution finished	Success		/tmp/extract-abcd	1	2011/11/15 12:41:34
chmod +x extract-abcd.sh	Start of job execution		Followed link after success			2011/11/15 12:41:34
chmod +x extract-abcd.sh	Job execution finished	Success			3	2011/11/15 12:41:34
extract-abcd.sh	Start of job execution		Followed link after success	/tmp/extract-abcd		2011/11/15 12:41:34

Figure 4–2 Execution result of J.1

When the Job had been running without problem a new ESE records has been created (see Figure 4–3 for comparison with Figure 4–1)

```
<section name="raw">
  <europeana:record xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:europeana="http://www.europeana.eu/schemas/ese/" xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/" _record_invalid="true" _record_invalid_reason="Missing license uri for
  rights. ">
    <dc:title>Abrus praecatorius</dc:title>
    <dc:description>herbarium sheet</dc:description>
    <dc:description/>
    <dc:type>PreservedSpecimen</dc:type>
    <dcterms:spatial/>
    <europeana:dataprovder>Botanic Garden and Botanical Museum Berlin-Dahlem</europeana:dataprovder>
    <europeana:isShownAt>http://ww2.bgbm.org/herbarium/view_biocase.cfm?SpecimenPK=140675</europeana:isShow
    nAt>
    <dc:identifier>Herbarium Berolinense - BGBM - B -W 13094 -00 1</dc:identifier>
    <europeana:rights>http://www.europeana.eu/portal/rr-f.html</europeana:rights>
    <dc:rights>The use of the data is allowed only for non-profit scientific use and for non-profit nature conservation
    purpose. The data base or part of it may only be used or copied by the written permission from the legal
    owner.</dc:rights>
    <dc:rights>Röpert, D. (Ed.) 2000 - (continuously updated): Digital specimen images at the Herbarium Berolinense
    (B).</dc:rights>
    <dc:source>Herbarium Berolinense</dc:source>
  </europeana:record>
</section>
```

Figure 4–3 Created ESE record with Pentaho



#### 4.2.1 Created folder structure

Figure 4–4 shows the structure of the directory that has been created with Pentaho. The first two folders name the content provider and a specific collection. The folders “oaiImported”, “log”, “extracted” and “ese” have been created with our Job. In “oaiImported” all records that have been imported in OAI are saved (see Figure 4–5). The “log” folder contains logfiles in .csv<sup>31</sup> format (see Figure 4–6) and “extracted” holds all “search\_response” files with ABCD records that have been extracted from the HIT database (see Figure 4–7).

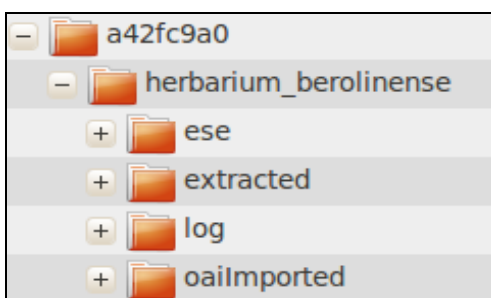


Figure 4–4 Structure of file directory

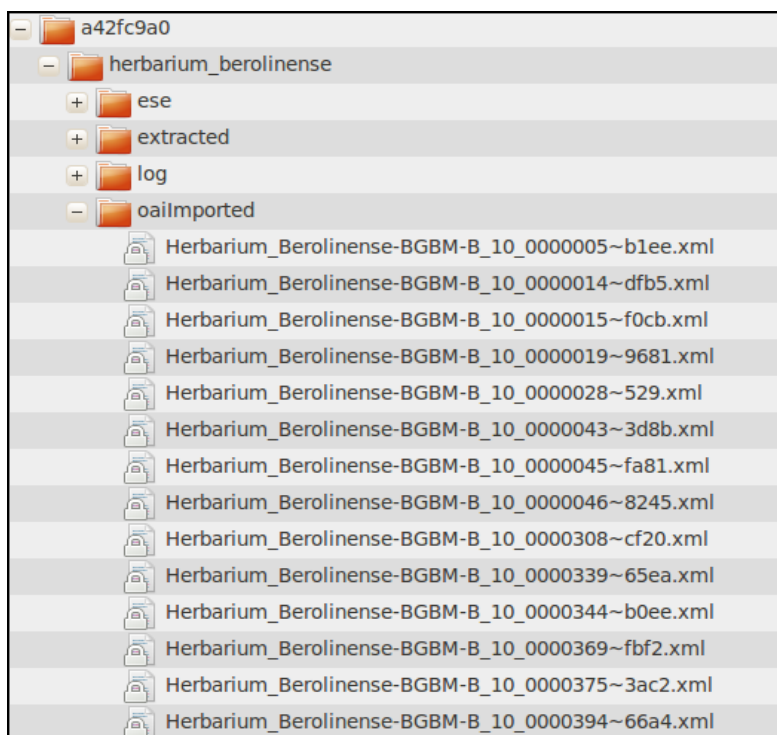


Figure 4–5 Folder “oaiImported” containing all in OAI imported records in XML format

<sup>31</sup> [http://en.wikipedia.org/wiki/Comma-separated\\_values](http://en.wikipedia.org/wiki/Comma-separated_values)

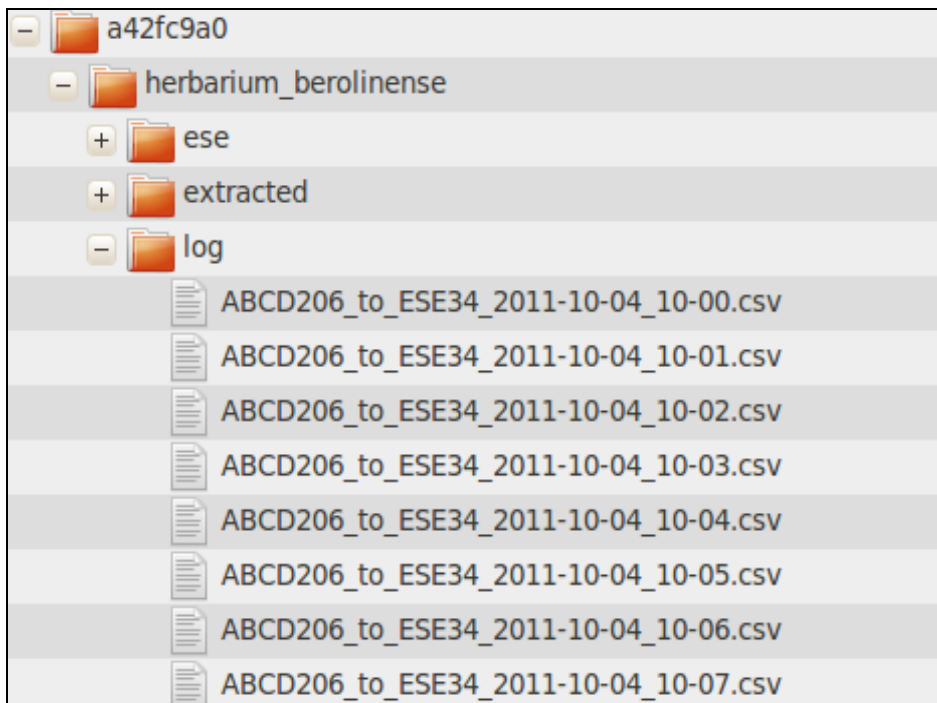


Figure 4–6 Folder “log” containing logfiles

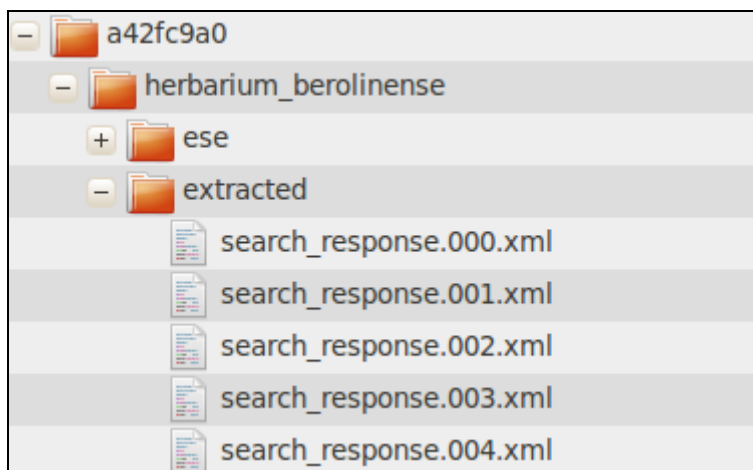


Figure 4–7 Folder “extracted” containing the ABCD records from the HIT database

Figure 4–8 shows an example .csv file from the folder “log” containing the reasons for incorrect ABCD records.

A	B	C	D	E	F	G	H				
1	Herbarium	Berolinense	BGBM	200154138	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
2	Herbarium	Berolinense	BGBM	200154141	N	The	unit	ID	>B		200154141<
3	Herbarium	Berolinense	BGBM	200154127	N	The	unit	ID	>B		200154127<
4	Herbarium	Berolinense	BGBM	200154128	N	The	unit	ID	>B		200154128<
5	Herbarium	Berolinense	BGBM	200154157	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
6	Herbarium	Berolinense	BGBM	200154160	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
7	Herbarium	Berolinense	BGBM	200154161	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
8	Herbarium	Berolinense	BGBM	200154152	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
9	Herbarium	Berolinense	BGBM	200154147	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
10	Herbarium	Berolinense	BGBM	200154142	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
11	Herbarium	Berolinense	BGBM	200129093	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
12	Herbarium	Berolinense	BGBM	200129124	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
13	Herbarium	Berolinense	BGBM	200129249	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
14	Herbarium	Berolinense	BGBM	200130032	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
15	Herbarium	Berolinense	BGBM	200154140	N	The	unit	ID	>B		200154140<
16	Herbarium	Berolinense	BGBM	200154141	N	The	unit	ID	>B		200154141<
17	Herbarium	Berolinense	BGBM	200154263	N	The	unit	ID	>B		200154263<
18	Herbarium	Berolinense	BGBM	200127112	N	The	unit	ID	>B		200127112<
19	Herbarium	Berolinense	BGBM	200142838	N	The	unit	ID	>B		200142838<
20	Herbarium	Berolinense	BGBM	200147494	N	The	unit	ID	>B		200147494<
21	Herbarium	Berolinense	BGBM	200147494	N	The	unit	ID	>B		200147494<
22	Herbarium	Berolinense	BGBM	200127957	N	The	unit	ID	>B		200127957<
23	Herbarium	Berolinense	BGBM	200127956	N	The	unit	ID	>B		200127956<
24	Herbarium	Berolinense	BGBM	200127808	N	The	unit	ID	>B		200127808<
25	Herbarium	Berolinense	BGBM	200096196	N	The	unit	ID	>B		200096196<
26	Herbarium	Berolinense	BGBM	200129133	N	The	unit	ID	>B		200129133<
27	Herbarium	Berolinense	BGBM	200154733	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
28	Herbarium	Berolinense	BGBM	200147497	N	The	unit	ID	>B		200147497<
29	Herbarium	Berolinense	BGBM	200154725	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
30	Herbarium	Berolinense	BGBM	200154728	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
31	Herbarium	Berolinense	BGBM	200128254	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
32	Herbarium	Berolinense	BGBM	200095101	N	The	unit	ID	>B		200095101<
33	Herbarium	Berolinense	BGBM	200154106	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
34	Herbarium	Berolinense	BGBM	200154107	N	Missing	license	uri	for	rights.	/opt/ht/4161387-bgbm-herbar/herbarium_berolinense/eseWithErrors//Herbarium_Bero
35	Herbarium	Berolinense	BGBM	200147490	N	The	unit	ID	>B		200147490<

Figure 4–8 .csv file in the “log” folder

#### 4.2.2 The .tmp directory

The third Transformation transformed the ABCD record into an ESE record. Furthermore in T.3 a shell script (see Figure 4–8) has been created and log files as well (see Figure 4–9) – both found in the .tmp directory.

```
#!/bin/bash
OUT="$2/extracted/$(echo -n "$3" | sed 's,.$gz$,.xml,')
test -d "$2/extracted" || mkdir "$2/extracted"
test -d "$2/ese" || mkdir "$2/ese"
test -d "$2/eseWithErrors" || mkdir "$2/eseWithErrors"
test -d "$2/log" || mkdir "$2/log"
if [ $? -ne 0 ]; then
    echo "You did not give me write permissions in $2"
    exit 1
fi
test "$1" -nt "$OUT" && gunzip <"$1" >"$OUT"
exit 0
```

Figure 4–9 Shell script extract-abcd.sh

```

1 2011/11/15 12:31:42 - Spoon - Logging goes to file:///tmp/spoon_648609a3-0f7d-11e1-b715-a180dbf1f30d.log
2 2011/11/15 12:31:49 - class org.pentaho.agilebi.platform.JettyServer - WebServer.Log.CreateListener localhost:10000
3 2011/11/15 12:31:49 - Version checker - OK
4 2011/11/15 12:31:52 - Spoon - Asking for repository
5 2011/11/15 12:31:53 - RepositoriesMeta - Reading repositories XML file: /home/ait112/.kettle/repositories.xml
6 2011/11/15 12:32:43 - Spoon - Save as...
7 2011/11/15 12:34:48 - T.1 Get Files from HIT DB - Dispatching started for transformation [T.1 Get Files from HIT DB]
8 2011/11/15 12:34:48 - T.1.1 From HIT DB AIT11.0 - Finished reading query, closing connection.
9 2011/11/15 12:34:48 - Spoon - The transformation has finished!!
10 2011/11/15 12:34:58 - T.1 Get Files from HIT DB - Dispatching started for transformation [T.1 Get Files from HIT DB]
11 2011/11/15 12:34:58 - T.1.1 From HIT DB AIT11.0 - Finished reading query, closing connection.
12 2011/11/15 12:34:58 - Spoon - The transformation has finished!!
13 2011/11/15 12:35:21 - T.2 adapt abcd load for ese transform - Dispatching started for transformation [T.2 adapt_abcd_load_for_ese_transform]
14 2011/11/15 12:35:21 - Spoon - The transformation has finished!!
15 2011/11/15 12:35:57 - T.3 ABCD206 to ESE34 - Dispatching started for transformation [T.3 ABCD206_to_ESE34]
16 2011/11/15 12:35:57 - Spoon - The transformation has finished!!
17 2011/11/15 12:36:45 - T.3 ABCD206 to ESE34 - Dispatching started for transformation [T.3 ABCD206_to_ESE34]
18 2011/11/15 12:36:45 - Spoon - The transformation has finished!!
19 2011/11/15 12:37:03 - T.3 ABCD206 to ESE34 - Dispatching started for transformation [T.3 ABCD206_to_ESE34]
20 2011/11/15 12:37:03 - Spoon - The transformation has finished!!
21 2011/11/15 12:37:32 - T.3 ABCD206 to ESE34 - Dispatching started for transformation [T.3 ABCD206_to_ESE34]
22 2011/11/15 12:37:32 - Spoon - The transformation has finished!!
23 2011/11/15 12:37:46 - T.3 ABCD206 to ESE34 - Dispatching started for transformation [T.3 ABCD206_to_ESE34]
24 2011/11/15 12:37:46 - Spoon - The transformation has finished!!
25 2011/11/15 12:37:57 - T.3 ABCD206 to ESE34 - Dispatching started for transformation [T.3 ABCD206_to_ESE34]
26 2011/11/15 12:37:57 - Spoon - The transformation has finished!!
27 2011/11/15 12:38:24 - Spoon - Save as...
28 2011/11/15 12:38:43 - Spoon - Spoon

```

Figure 4–10 Created Log file with Pentaho

There also exists a logfile for the shell script in the .tmp directory (see Figure 4–10).

```

2011/11/15 12:41:31 - J.1 Biocase Harvest to ESE - Start of job execution
2011/11/15 12:41:31 - J.1 Biocase Harvest to ESE - Starting entry [Get Files from HIT DB]
2011/11/15 12:41:32 - Get Files from HIT DB - Dispatching started for transformation [Get Files from HIT DB]
2011/11/15 12:41:32 - From HIT DB.0 - Finished reading query, closing connection.
2011/11/15 12:41:32 - From HIT DB.0 - Finished processing (I=1, O=0, R=0, W=1, U=0, E=0)
2011/11/15 12:41:32 - get parameters, get directory.0 - Finished processing (I=1, O=0, R=1, W=1, U=0, E=0)
2011/11/15 12:41:32 - drop json.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2011/11/15 12:41:32 - set file pattern search_reponse.*gz.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2011/11/15 12:41:32 - build filesystem path.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2011/11/15 12:41:34 - Get File Names.0 - Finished processing (I=0, O=0, R=1, W=6084, U=0, E=0)
2011/11/15 12:41:34 - Select result.0 - Finished processing (I=0, O=0, R=6084, W=6084, U=0, E=0)
2011/11/15 12:41:34 - Copy rows to result.0 - Finished processing (I=0, O=0, R=6084, W=6084, U=0, E=0)
2011/11/15 12:41:34 - J.1 Biocase Harvest to ESE - Starting entry [create extract-abcd.sh]
2011/11/15 12:41:34 - J.1 Biocase Harvest to ESE - Starting entry [chmod +x extract-abcd.sh]
2011/11/15 12:41:34 - chmod +x extract-abcd.sh - Running on platform : Linux
2011/11/15 12:41:34 - chmod +x extract-abcd.sh - Executing command : /tmp/kettle_c540a6f4-0f7e-11e1-b715-a180dbf1f30dshell
2011/11/15 12:41:34 - J.1 Biocase Harvest to ESE - Starting entry [extract-abcd.sh]
2011/11/15 12:41:34 - create extract-abcd.sh - Found 6084 previous result rows
2011/11/15 12:41:34 - create extract-abcd.sh - Running on platform : Linux
2011/11/15 12:41:34 - create extract-abcd.sh - Executing command : /tmp/extract-abcd.sh /opt/hit/4161387-bgbm-herbar/herbarium_berolinense/
search_response.000.gz /opt/hit/4161387-bgbm-herbar/herbarium_berolinense search_response.000.gz gz file Y N Y Y 2011/08/18 11:17:11.000 8318
2011/11/15 12:41:34 - create extract-abcd.sh - Command /tmp/extract-abcd.sh /opt/hit/4161387-bgbm-herbar/herbarium_berolinense/
search_response.000.gz /opt/hit/4161387-bgbm-herbar/herbarium_berolinense search_response.000.gz gz file Y N Y Y 2011/08/18 11:17:11.000 8318 has
finished
2011/11/15 12:41:34 - create extract-abcd.sh - Running on platform : Linux
2011/11/15 12:41:34 - create extract-abcd.sh - Executing command : /tmp/extract-abcd.sh /opt/hit/4161387-bgbm-herbar/herbarium_berolinense/
search_response.001.gz /opt/hit/4161387-bgbm-herbar/herbarium_berolinense search_response.001.gz gz file Y N Y Y 2011/08/18 11:17:24.000 6936
2011/11/15 12:41:34 - create extract-abcd.sh - Command /tmp/extract-abcd.sh /opt/hit/4161387-bgbm-herbar/herbarium_berolinense/
search_response.001.gz /opt/hit/4161387-bgbm-herbar/herbarium_berolinense search_response.001.gz gz file Y N Y Y 2011/08/18 11:17:24.000 6936 has
finished
2011/11/15 12:41:34 - create extract-abcd.sh - Running on platform : Linux
2011/11/15 12:41:34 - create extract-abcd.sh - Executing command : /tmp/extract-abcd.sh /opt/hit/4161387-bgbm-herbar/herbarium_berolinense/
search_response.002.gz /opt/hit/4161387-bgbm-herbar/herbarium_berolinense search_response.002.gz gz file Y N Y Y 2011/08/18 11:17:41.000 12541
2011/11/15 12:41:35 - create extract-abcd.sh - Command /tmp/extract-abcd.sh /opt/hit/4161387-bgbm-herbar/herbarium_berolinense/
search_response.002.gz /opt/hit/4161387-bgbm-herbar/herbarium_berolinense search_response.002.gz gz file Y N Y Y 2011/08/18 11:17:41.000 12541 has
finished

```

Figure 4–11 Log file for the shell script extract-abcd.sh

## Glossar

<b>ABCD</b>	<b>Access to Biological Collection Data</b> An evolving comprehensive standard for the access to and exchange of data about specimens and observations. <a href="http://wiki.tdwg.org/twiki/bin/view/ABCD/AbcdPrimer">http://wiki.tdwg.org/twiki/bin/view/ABCD/AbcdPrimer</a>
<b>BioCASE</b>	<b>Biological Collection Access Service for Europe</b> A transnational network of biological collections of all kinds. <a href="http://www.biocase.org/">http://www.biocase.org/</a>
<b>CSV</b>	<b>Comma Separated Values</b> Stores tabular data in plain text form. <a href="http://en.wikipedia.org/wiki/Comma-separated_values">http://en.wikipedia.org/wiki/Comma-separated_values</a>
<b>DC</b>	<b>Dublin Core</b> A vocabulary of fifteen properties for use in resource description. <a href="http://www.dublincore.org/documents/">http://www.dublincore.org/documents/</a>
<b>ETL</b>	<b>Extract-Transform-Load</b> 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." <a href="http://www.stylusstudio.com/etl/">http://www.stylusstudio.com/etl/</a>
<b>ESE v3.4</b>	The <b>Europeana Semantic Elements</b> specification of the Europeana Portal ( <a href="http://www.europeana.eu">www.europeana.eu</a> ) <a href="http://www.europeana.eu/schemas/ese/ESE-V3.4.xsd">http://www.europeana.eu/schemas/ese/ESE-V3.4.xsd</a>
<b>FTP</b>	" <b>File Transfer Protocol</b> is a standard protocol used to exchange and manipulate files over an Internet Protocol computer network, such as the Internet." <a href="http://en.wikipedia.org/wiki/File_Transfer_Protocol">http://en.wikipedia.org/wiki/File_Transfer_Protocol</a> [Stand: 14.07.2009]
<b>GBIF</b>	<b>Global Biodiversity Information Facility</b> aim: encourage free and open access to biodiversity data, via the Internet. <a href="http://www.gbif.org/">http://www.gbif.org/</a>
<b>HIT</b>	<b>Harvesting Index Toolkit</b> An open source Java-based web application developed by the GBIF Secretariat to manage biodiversity data harvesting and quickly build indexes of harvested data. <a href="http://code.google.com/p/gbif-indexingtoolkit/">http://code.google.com/p/gbif-indexingtoolkit/</a>
<b>JavaScript</b>	<i>THE</i> scripting language of the Web. <a href="http://www.w3schools.com/js/default.asp">http://www.w3schools.com/js/default.asp</a>
<b>Json</b>	<b>JavaScript Object Notation</b> A lightweight data-interchange format. <a href="http://www.json.org/">http://www.json.org/</a>

---

<b>MySQL</b>	MySQL is a relational database management system. <a href="http://www.w3schools.com/PHP/php_mysql_intro.asp">http://www.w3schools.com/PHP/php_mysql_intro.asp</a>
<hr/>	
<b>PHP</b>	<b>PHP: Hypertext Preprocessor (Personal Home Page)</b> "PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML." <a href="http://www.php.net/">http://www.php.net/</a>
<hr/>	
<b>RegExp</b>	<b>Regular Expression</b> "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. <a href="http://www.regular-expressions.info/">http://www.regular-expressions.info/</a>
<hr/>	
<b>Shell Script</b>	A series of commands written in plain text files <a href="http://www.freeos.com/guides/lsst/">http://www.freeos.com/guides/lsst/</a>
<hr/>	
<b>SQL</b>	The <b>Structured Query Language</b> is a database language designed for managing data in a relational database management system. <a href="http://www.w3schools.com/SQL/sql_intro.asp">http://www.w3schools.com/SQL/sql_intro.asp</a>
<hr/>	
<b>SSH</b>	<b>Secure Shell</b> 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. <a href="http://en.wikipedia.org/wiki/Secure_Shell">http://en.wikipedia.org/wiki/Secure_Shell</a>
<hr/>	
<b>wildcard</b>	A character that may be substituted for any of a defined subset of all possible characters. <a href="http://en.wikipedia.org/wiki/Wildcard_character">http://en.wikipedia.org/wiki/Wildcard_character</a>
<hr/>	
<b>XML</b>	"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." <a href="http://www.w3.org/XML/">http://www.w3.org/XML/</a> [Stand: 14.07.2009]
<hr/>	
<b>XPath</b>	XPath is used to navigate through elements and attributes in an XML document. <a href="http://www.w3schools.com/xpath/">http://www.w3schools.com/xpath/</a>
<hr/>	
<b>XSL</b>	<b>EXtensible Stylesheet Language</b> A stylesheet language for XML. <a href="http://www.w3schools.com/xsl/">http://www.w3schools.com/xsl/</a>

---

## I. List of Figures

Figure 1–1 The ESE Schema .....	7
Figure 1–2 Pentaho Welcome Screen with Repository Connection window .....	8
Figure 1–3 Repository Connection window.....	8
Figure 2–1 Categories of Transformation steps .....	9
Figure 2–2 Deleting an established hop .....	10
Figure 2–3 Control icons of a transformation .....	10
Figure 2–4 Control icons of a Job .....	10
Figure 4–1 One Unit of an ABCD record .....	72
Figure 4–2 Execution result of J.1 .....	72
Figure 4–3 Created ESE record with Pentaho .....	72
Figure 4–4 Structure of file directory .....	73
Figure 4–5 Folder “oaiImported” containing all in OAI imported records in XML format.....	73
Figure 4–6 Folder “log” containing logfiles.....	74
Figure 4–7 Folder “extracted” containing the ABCD records from the HIT database .....	74
Figure 4–8 .csv file in the “log” folder .....	75
Figure 4–9 Shell script extract-abcd.sh .....	75
Figure 4–10 Created Log file with Pentaho .....	76
Figure 4–11 Log file for the shell script extract-abcd.sh.....	76



## II. List of Pentaho Figures

Figure J.1 Job in Pentaho including three transformations.....	11
Figure T.1 Get_Files_from_HIT_DB Transformation .....	12
Figure T.1.1 Table Input .....	13
Figure T.1.1a Define Table Input.....	13
Figure T.1.1b Database connection .....	14
Figure T.1.1c Fields “uddi_key”, “name” and “parameters_as_json” from the HIT DB .....	15
Figure T.1.2 Json Input connected to Table Input .....	15
Figure T.1.2a File section – Json Input.....	15
Figure T.1.2b Content section – Json Input .....	16
Figure T.1.2c Field section – Json Input .....	16
Figure T.1.3 Connection to the third step Select values – drop json .....	17
Figure T.1.3a Select Values – Remove “parameters_as_json” .....	17
Figure T.1.4 Adding the “Add Constants” step.....	18
Figure T.1.4a Add the constants “pattern” and “path” .....	18
Figure T.1.5 Adding the “Replace in String” step .....	19
Figure T.1.5a Replace in String.....	19
Figure T.1.6 Adding the “Get File Names” step.....	20
Figure T.1.6a File section of the step Get File Names .....	20
Figure T.1.6b Filters section of the step Get File Names .....	21
Figure T.1.7 Adding another “Select Values” step .....	21
Figure T.1.7a Get fields to select .....	22
Figure T.1.8 Copy rows to result .....	22
Figure T.1.8a Rows to display during preview .....	23
Figure J.2 The Pentaho Job .....	24
Figure T.2 The “adapt_ABCD_load_for_ese_transform” transformation.....	24
Figure T.2.1 The “Get rows from result” step.....	25
Figure T.2.1a “Get rows from result” menu .....	25



Figure T.2.2 Adding the “Replace in string” step.....	25
Figure T.2.2a Defining search and replace values in the “Replace in string” step .....	25
Figure T.2.3 Adding the “Select values” step .....	26
Figure T.2.3a Select and rename values .....	26
Figure J.3 The Job with the current Transformation marked red.....	27
Figure T.3 The ABCD206_to_ESE34 Transformation.....	27
Figure T.3.1 The two input steps “Input filenames” and “Test data” .....	28
Figure T.3.1a Chosen fields in the “Get rows from result” step .....	29
Figure T.3.2a Generated rows in the menu of the “Generate rows” step.....	29
Figure T.3.3 Adding the step “Load file content in memory” .....	30
Figure T.3.3a File section of the “Load file content in memory” step.....	30
Figure T.3.3b Fields section of “Load file content in memory” .....	31
Figure T.3.4 Adding the “Get XML data” step .....	31
Figure T.3.4a File section of “Get XML data” .....	32
Figure T.3.4b Content section of “Get XML data” .....	32
Figure T.3.4c Fields section of “Get XML data” .....	33
Figure T.3.5 Adding the “Add constants” step .....	33
Figure T.3.5a Value of the “Add constants” step.....	34
Figure T.3.6 Adding the “XSL Transformation” step .....	35
Figure T.3.6a Settings of the XSL Transformation.....	35
Figure T.3.6b The Advanced section of XSL Transformation .....	36
Figure T.3.7 Adding another “Select values” step .....	37
Figure T.3.7a “Select values” step – removing “abcdXML” and “abcd2EuropeanaXSL” .....	37
Figure T.3.8 Adding another “Add constants” step .....	38
Figure T.3.8a Adding the result field “recordOK” .....	38
Figure T.3.9 “Filter rows” step with two output steps .....	38
Figure T.3.9a The “Filter rows” step with the condition.....	39
Figure T.3.10 Getting the error filename from UnitID with JavaScript .....	40
Figure T.3.11 Getting the filename from UnitID with JavaScript.....	40

Figure T.3.12 Adding two "Text file output" steps .....	42
Figure T.3.12a File section of "Save erroneous ESE records" .....	43
Figure T.3.12b Content section of "Save erroneous ESE records" .....	43
Figure T.3.12c Fields section of "Save erroneous ESE records" .....	44
Figure T.3.13a File section of "Save ESE records" .....	44
Figure T.3.13b Content section of "Save ESE records" with default values .....	45
Figure T.3.13c Fields section of "Save ESE records" .....	45
Figure T.3.14 Adding a "Set field value to constant" step to our Transformation.....	46
Figure T.3.14a Defining the constant "Y" .....	46
Figure T.3.15 Adding another "Add constant value" step .....	47
Figure T.3.15a Setting errorReason "OK" .....	47
Figure T.3.16 Adding an "Add constants" step to T.3.3.....	48
Figure T.3.16a Constant fields added to the Transformation.....	48
Figure T.3.17 Adding a "Select values" step to remove the input file .....	49
Figure T.3.17a Removing the second input file.....	50
Figure T.3.18 Adding a "Set field value to constant" step.....	50
Figure T.3.18a "set NOT OK" with the value "false" .....	51
Figure T.3.19 Adding a "Get data from XML" step to find the error reason.....	51
Figure T.3.19a File section of "Get error Reason" .....	52
Figure T.3.19b Content section of "Get error Reason" .....	53
Figure T.3.19c Fields section of "Get error Reason" with field "errorReason" .....	53
Figure T.3.20 Adding "Select result" and connecting the steps.....	54
Figure T.3.20a Selected fields for result .....	54
Figure T.3.21 Adding "Modified Java Script value" to create logfilename with date.....	55
Figure T.3.21 a Creating logfilename with date with Java Script .....	55
Figure T.3.22 Adding another "Text file output" step to create the logfiles.....	56
Figure T.3.22a File section of "Log: ABCD206_to_ESE34_<date>.csv .....	57
Figure T.3.22b Content section of the "Text file output" step that creates a logfile .....	58
Figure T.3.22c Fields section of "Log: ABCD206_to_ESE34_<data>.csv.....	58

Figure T.3.23 Copy rows to result ..... 59

Figure J.4 The Pentaho Job (II) ..... 60

Figure J.1a First two steps in the Pentaho Job..... 60

Figure J.1b Scheduling a Job ..... 61

Figure J.1.T.1a Transformation specification ..... 61

Figure J.1.T.1b Parameters of T.1 ..... 62

Figure J.1.1 Adding the “Write to file” step to the job..... 62

Figure J.1.1a Settings of the “Write to file” step ..... 63

Figure J.1.2 Adding an “Shell” step to the job ..... 63

Figure J.1.2a General section of the step ..... 64

Figure J.1.2b Script of “Shell” ..... 64

Figure J.1.3 Add another “Shell” step to our Job..... 65

Figure J.1.3a General section for “extract-abcd.sh” ..... 66

Figure J.1.3b Script of “extract-abcd.sh” ..... 66

Figure J.1.T.2 Adding the “adapt\_abcd\_load\_for\_ese\_transform” Transformation..... 67

Figure J.1.T.2a Transformation specification for “adapt\_abcd\_load\_for\_ese\_transform” ..... 67

Figure J.1.T.2b Advanced settings for “adapt\_abcd\_load\_for\_ese\_transform” ..... 68

Figure J.1.T.2c Parameters of “adapt\_abcd\_load\_for\_ese\_transform” ..... 68

Figure J.1.T.3 Adding the “ABCD206\_to\_ESE34” transformation ..... 69

Figure J.1.T.3a Transformation specification ..... 69

Figure J.1.T.3b Transformation specification for “ABCD206\_to\_ESE34” ..... 70

Figure J.1.T.3c Advanced section of “ABCD206\_to\_ESE34” transformation ..... 70

Figure J.1c Success ..... 70