Pentaho Data Integration
Previously Kettle
Version 3.0

Pentaho Data Integration version 3.0

Making a fresh start

Key differences with version 2.5.0

List of changes on November 14th 2007

Compiled by Matt Casters, mcasters (at) pentaho.org

Send additional changes you found to this address.

Index

1. Changes summary

1.1. Preface

Creating Pentaho Data Integration version 3.0 was a challenge in all meanings of the word. When you start the new and improved Spoon version, you will notice the new icons look. However, the changes that went on in the core API have been much bigger than what meets the eye on the first glance.

At a certain point during development you come to the realization that certain aspects of the existing software design is not turning out quite the way you wanted or envisioned at the start. This realization is at the core of most evolutions in software development and certainly also for PDI 3.0. We tried to get rid of every bottlenecks and inconsistency we could think of. Now that we in fact did get rid of those, we believe that in the future we'll be able to evolve even faster than we already do.

In this release we've again managed to attract more and more people in the community to help out with coding, documentation, translation, packaging (w32 and OSX installers), bug reporting and a lot more. This document was written as a special "thank you" note to all those people involved and to keep everyone informed about the enormous progress we are making.

1.2. Overview

These are the most notable changes that have been made:

- New core API
  - Split of data and meta-data offering greater flexibility and performance
  - Centralized data conversion API with accurate and enhanced error reporting
  - Simplified and unified preview and debug architecture (with listeners)
  - Advanced data storage engine
    - Normal storage: native data types (as before)
    - Binary string storage: for lazy conversion (see below)
    - Indexed storage: references a table of contents. The data is a simple integer index.
  - Improved step and job plug-in system with annotations:
    - http://wiki.pentaho.org/display/EAI/Annotated+step+plugin+development

- Improved transformation preview and debug
  - Preview with “Get more rows” functionality.
  - Transformation start, pause, resume, stop
  - Debugger with conditions (breakpoint pause)

- New high performance text file handling
  - Lazy conversion helps to prevent String data conversion when it's not really needed
  - New high performance “CSV Input” step with Non-blocking IO (NIO) support
  - New high performance “Fixed Input” step with NIO and parallel reading support

- Revised Spoon GUI
  - New set of icons with consistent color scheme.
  - First steps towards full XUL support (Menu, Tool bar, ...)
  - Created more possibilities for Spoon customization (OEM) of look and feel

- Mappings on steroids
  - Allows zero or more input streams
  - Allows zero or more output streams
  - Can be parameterized using variables
  - Supports global field renames on input and output

- Other new Steps
  - Access Input : reads MS-Access MDB files directly.
  - LDAP Input: reads information from an LDAP server
  - Append streams: append 2 streams in an ordered way
  - Abort: abort a transformation in case one or more rows are read
  - Regex Evaluation : evaluate regular expressions
  - Mondrian Input : read data from a Mondrian server using MDX
  - Closure generator: generates transitive closured for use with Mondrian.
  - Get files rows count : count the number of rows in files

- New Job entries
  - XSL transformation : to transform XML documents
  - Success : to force success on a job after failure
  - XSD validator : validate an XML document using XSD
  - DTD validator : validate an XML document using DTD
  - Write to log file : write a message in the log file
  - Copy files : copies one or more sets of files from one location to another
  - Put a file with FTP
  - Unzip files

- Slave server improvements
  - Remote execution of jobs
  - Improved partitioning and re-partitioning support in a clustered environment
  - Added dynamic partition sizing support (number of steps per slave)

- Miscellaneous
  - Hundreds of issues fixed and features implemented (400+)
  - A new translator GUI (French translation is now complete)
  - A new software package for our OSX users
  - Put a new bug tracking system into action: http://jira.pentaho.org/browse/PDI
  - Adding more and more content to our PDI Wiki documentation site.
  - ...

2. General changes

2.1. New core API

2.1.1. Data and meta-data split

One of the notable changes is a redesign of the internal data engine.
More to the point, we now have a strict separation of data and meta-data.

The main reason for doing so was the reduction of object allocation and also to allow us to extend the meta-data in this release and in the future without even the slightest performance impact.
We anticipated a performance gain here and there, and initial test-code gave us a 15-20% increase in performance to hope for. What we have seen is a performance gain that is quite a bit higher than that. To measure the performance gain we made a library of test-cases to run performance and regression tests again version 2.5 code. The result with comparison (speedup calculation) is posted here.

One of the nice things about the code changes is that although it will break the API, it's a Good Think(TM) for the project in the long run. It will give us breathing room to keep innovating in the future:

- Select Values : x9 (Random select), x6 (Delete field), x5.5 (Change metadata)
- Calculations: between x1.8 and 3.6 faster
- Add sequence : up to 3x faster
- Table Output : 15% faster up to x2
- Add constant values: x5
- Filter rows: x1.5
- Row generator: x2.5 - x3 (up to 1.2M empty rows/s)
- Sort rows: x1.15 - x5
- Stream Lookup: x1.24 - x1.36 (and this step already got some serious tuning in the past)
- Table Input: x1.2
- Text File Input: x1.6 - x3.4
- Text File Output: x1.75 – x2.9
- ...

On top of that, memory usage should have been reduced as well.

Besides this we also cleaned up the data conversion algorithms. This has consequences in certain extreme cases. For a complete list of things to look out for during migration, go to our migration guide:

http://wiki.pentaho.org/display/EAI/Pentaho+Data+Integration+3.0+migration+guide

2.1.2. Lazy conversion

In the specific case where we're reading text files and writing data back to text files, a lot of work is being done for nothing. First we convert data during file read:

- byte[] (in a file) to Java String (UTF-8)
- String to Date, String to Integer, String to Number, etc. using a certain mask

Then we do the exact opposite again when we write data back to file:

- Integer to String, Date to String, Number to String, etc using a certain mask
- Java String (UTF-8) back to byte[]

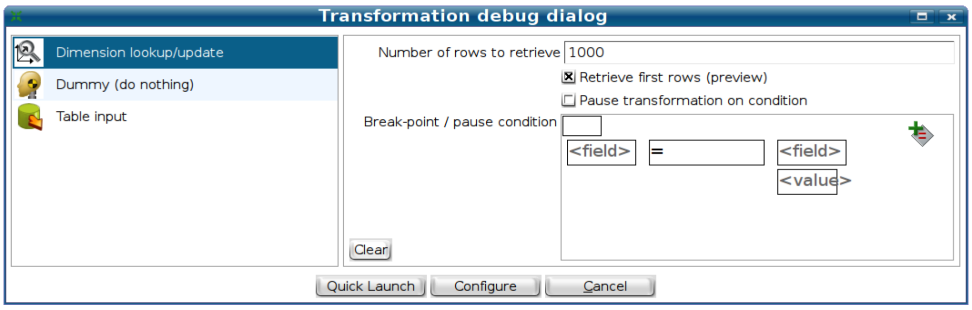
These conversions are not trivial since they take into account code-page conversions, locality, masks, decimal point and grouping symbols and a lot more. As such they consume a lot of CPU.

So we used the concept of "Lazy Conversion" in PDI 3.0. Lazy conversion delays conversion of data as long as possible with the hope it might never occur at all. Obviously, in the case where you read from a text-file and write back to another one in the same format, conversion never occurs. In that particular case, we read bytes and dump them to file again without ever looking at the actual content. (unless we have to in case the conversion formats are different for example)

See also: http://www.ibridge.be/?p=63 for more details and http://www.ibridge.be/?p=78 for a use-case.

2.2. Improved preview and debug

If you select one or more steps and select preview from the tool bar or menu, you will be presented with this transformation debug dialog:

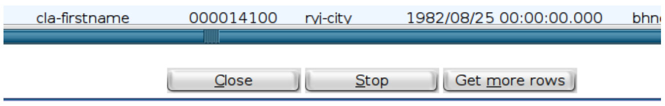

Transformation debug dialog

Dimension lookup/update
Dummy (do nothing)
Table input

Number of rows to retrieve: 1000
[✓ Retrieve first rows (preview)]
Pause transformation on condition
Break-point / pause condition: <field> = <field> <value>

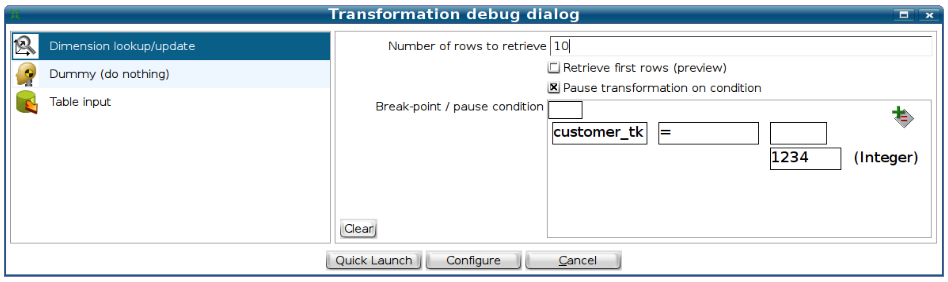
A configurable default number of rows to preview will be set on the selected steps. Hitting the “Quick launch” button will start the transformation in preview mode.

When the rows to preview are found for a certain step, they will be immediately shown in a dialog. You can there and then opt to get more rows to preview:


cla-firstname 000014100 ryj-city 1982/08/25 00:00:00.000 bhn

If you just close the dialog, the transformation will be in a paused state and can then be resumes from the transformation log window. (getting more rows to preview)

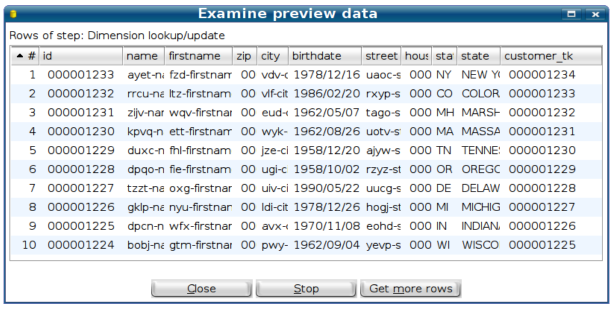
If you use the debug button, we will set different parameters on the selected steps:


Transformation debug dialog

Dimension lookup/update
Dummy (do nothing)
Table input

Number of rows to retrieve: 10
Retrieve first rows (preview)
[✓ Pause transformation on condition]
Break-point / pause condition: customer_tk = 1234 (Integer)

The condition can then be used to pause the transformation when a certain condition is met. In the example above we want to pause the transformation if we retrieve technical key “1234” in a dimension. Not only that, we want to see the last 10 records that passed through the step. When this record is found, it will be shown together with the other 9 requested in reversed order:


Examine preview data

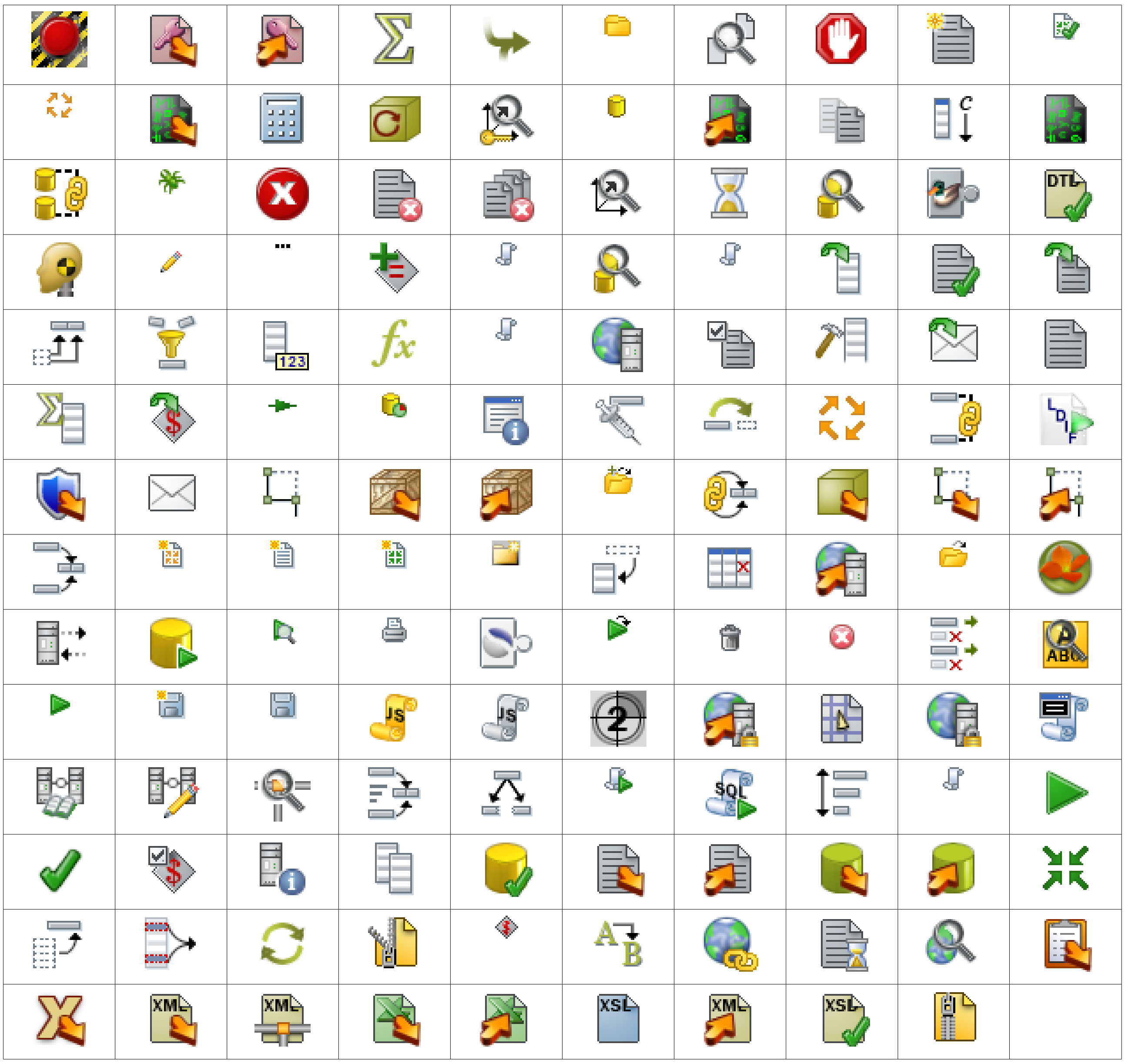
Rows of step: Dimension lookup/update

#	id	name	firstname	zip	city	birthdate	street	hous	sta	state	customer_tk
1	000001233	ayet-na	fzd-firstnam	00	vdv-c	1978/12/16	uaoc-s	000	NY	NEW YO	000001234
2	000001232	rrcu-na	ltz-firstnam	00	vlf-cit	1986/02/20	rxyp-s	000	CO	COLOR	000001233
3	000001231	zijv-nar	wqv-firstnar	00	eud-c	1962/05/07	tago-s	000	MH	MARSH	000001232
4	000001230	kpvq-n	ett-firstnam	00	wyk-	1962/08/26	uotv-s	000	MA	MASSA	000001231
5	000001229	duxc-n	fhl-firstnam	00	jze-ci	1958/12/20	ajyw-s	000	TN	TENNE:	000001230
6	000001228	dpqo-n	fie-firstnam	00	ugi-c	1958/10/02	rzyz-st	000	OR	OREGC	000001229
7	000001227	tzzt-na	oxg-firstnar	00	uiv-ci	1990/05/22	uucg-s	000	DE	DELAW	000001228
8	000001226	gklp-na	nyu-firstnar	00	ldi-cit	1978/12/26	hogj-st	000	MI	MICHIG	000001227
9	000001225	dpcn-n	wfx-firstnar	00	avx-c	1970/11/08	eohd-s	000	IN	INDIAN.	000001226
10	000001224	bobj-na	gtm-firstnar	00	pwy-	1962/09/04	yevp-s	000	WI	WISCO	000001225

2.3. Revised Spoon GUI

2.3.1. New icons

The excellent icons below where created by the Pentaho Pixel Manager and are the first thing you notice when you create new transformations or jobs or re-open your existing files.



2.3.2. Customization of Spoon

Version 3.0 of PDI allows you to configure Spoon to your taste. This was mostly done to make it easier to OEM the GUI elements. To do this, it must be possible to change icons, messages, colors and fonts without the need for any change to the existing code.

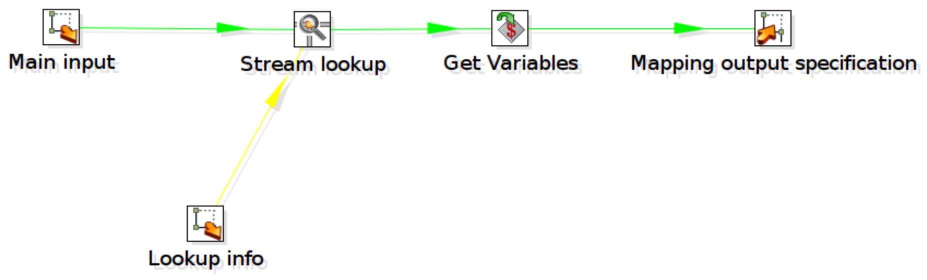
One of the notable changes in directions we took was that we started to use XUL (http://www.mozilla.org/projects/xul/) to drive our menus and tool bars. This allows people to inject their own menu-items and create their own tabs in Spoon.

For those that want to go even further we created hidden XML files (inside the kettle jars) to change a lot of aspects of the software behavior.

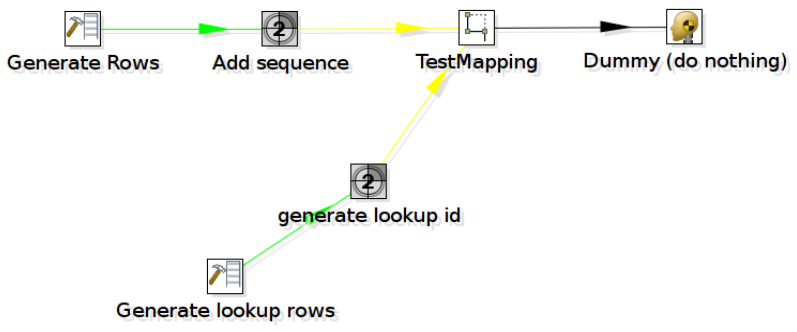
2.4. Mappings on steroids

Mappings in the past where limited in use. We increased the number of possible use-cases by allowing the mappings to be a lot more complex than before. At the same time we added various features to make usability better than before.

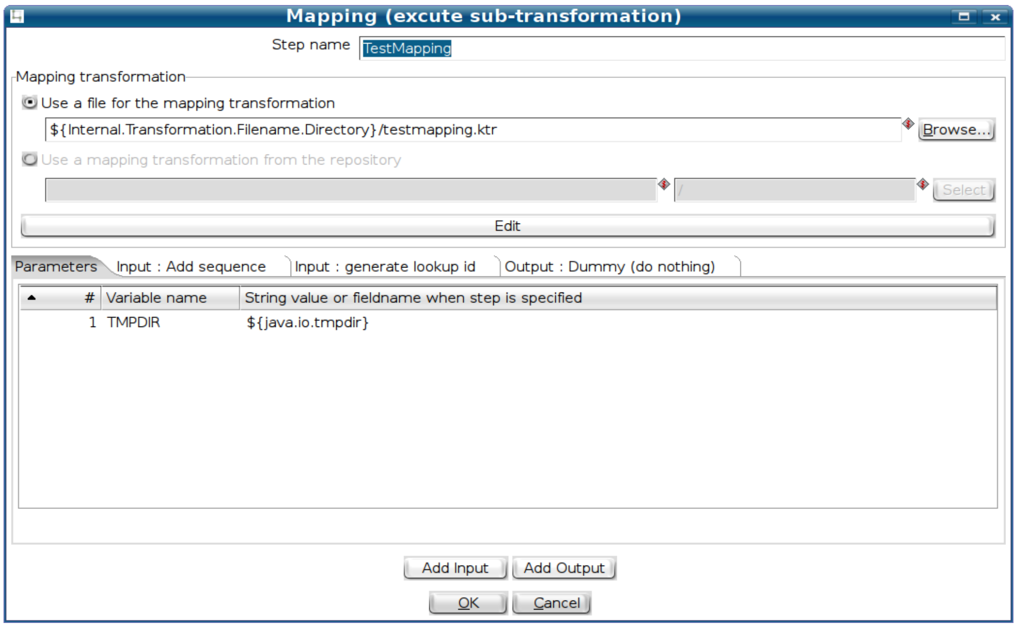
For example, this mapping has 2 inputs and a single output:



The mapping step to execute the transformation looks like this:



This is handled by the following mapping dialog:



Step name: TestMapping

Mapping transformation

Use a file for the mapping transformation: ${Internal.Transformation.Filename.Directory}/testmapping.ktr

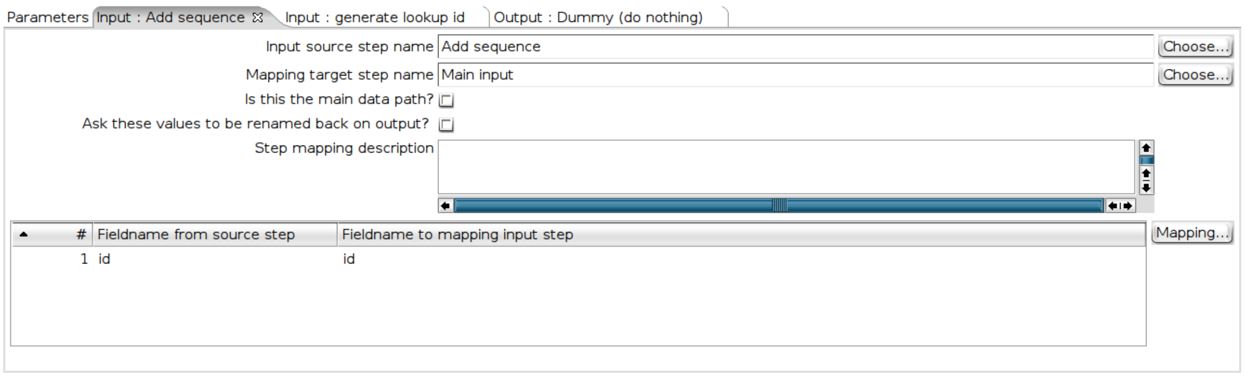
Use a mapping transformation from the repository

Tabs: Parameters | Input : Add sequence | Input : generate lookup id | Output : Dummy (do nothing)

#	Variable name	String value or fieldname when step is specified
1	TMPDIR	${java.io.tmpdir}

As you can see, there is a single tab created for every input or output stream you add and a single one to set the parameters on the mapping (sub-transformation).

The input and output tabs have several helper buttons to allow you to set up the mapping step as easily as possible:



Tabs: Parameters | Input : Add sequence | Input : generate lookup id | Output : Dummy (do nothing)

Input source step name: Add sequence

Mapping target step name: Main input

Is this the main data path? ☐

Ask these values to be renamed back on output? ☐

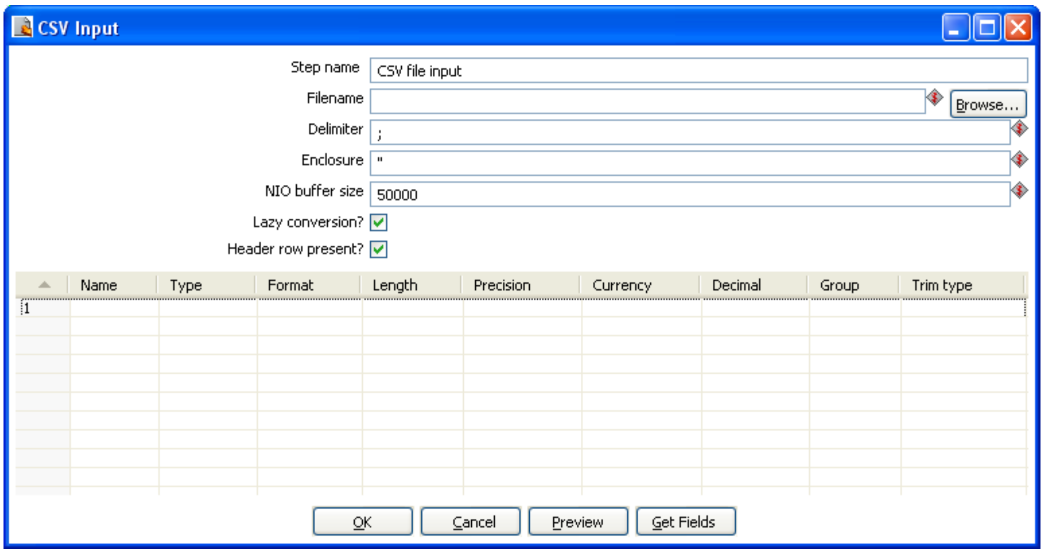
Step mapping description:

#	Fieldname from source step	Fieldname to mapping input step
1	id	id

2.5. New steps

Below is an overview of the new steps that were added. For the full details on these steps, please consult the Spoon user guide documentation.

2.5.1. CSV Input

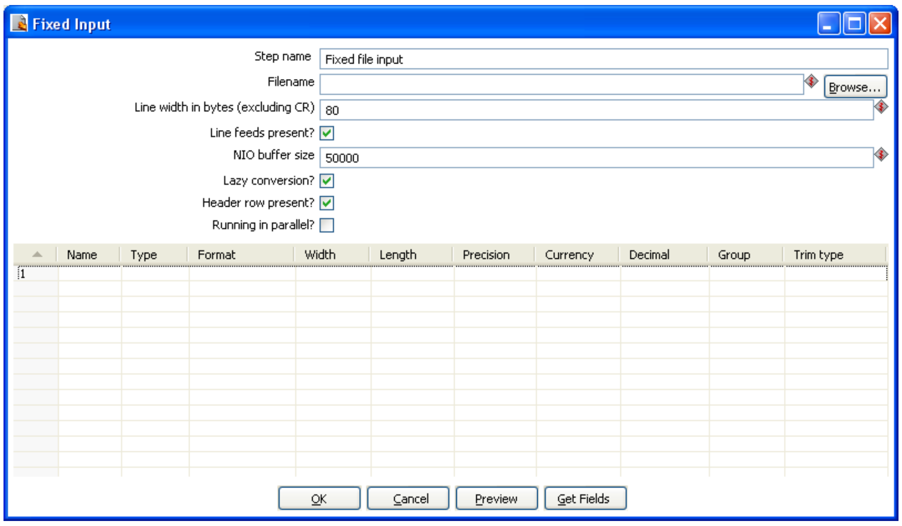


Step name	CSV file input
Filename	
Delimiter	;
Enclosure	"
NIO buffer size	50000
Lazy conversion?	✓
Header row present?	✓

	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1									

This step provides the ability to read data from a delimited file. It conforms to this CSV “standard” as much as possible: http://en.wikipedia.org/wiki/Comma-separated_values

2.5.2. Fixed Input

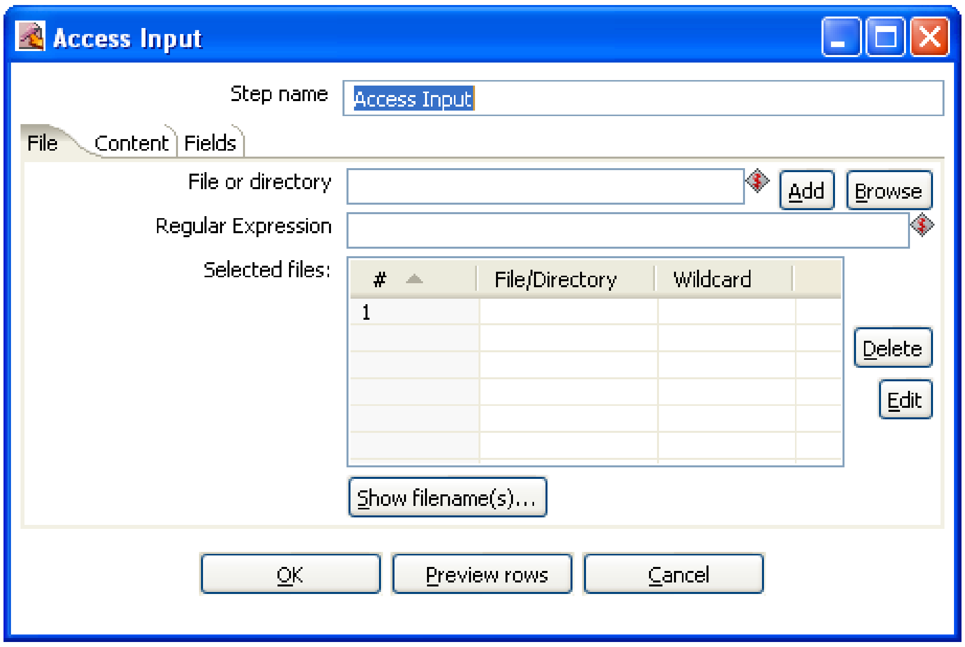


Step name	Fixed file input
Filename	
Line width in bytes (excluding CR)	80
Line feeds present?	✓
NIO buffer size	50000
Lazy conversion?	✓
Header row present?	✓
Running in parallel?	☐

	Name	Type	Format	Width	Length	Precision	Currency	Decimal	Group	Trim type
1										

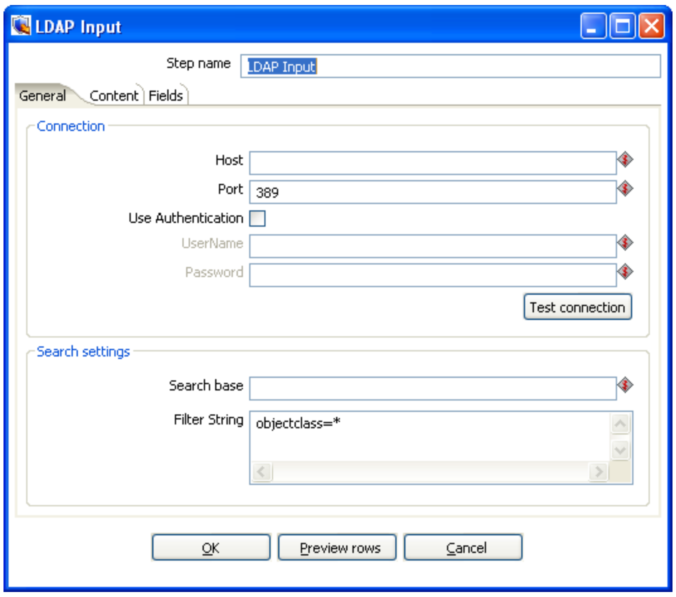
This step is used to read data from a fixed width file.

2.5.3. Access Input



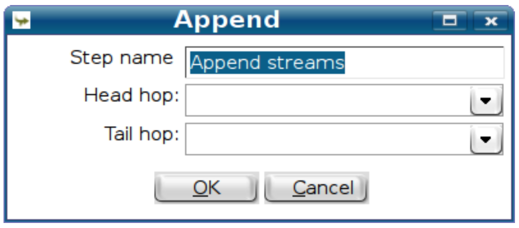
This step reads MS-Access MDB files directly.

2.5.4. LDAP Input



Reads information from an LDAP server

2.5.5. Append streams


Append

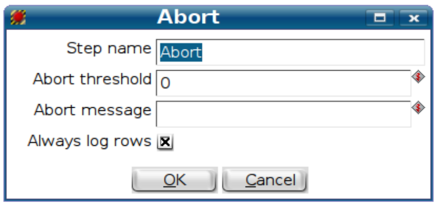
Step name: Append streams
Head hop:
Tail hop:

OK Cancel

Appends 2 streams in an ordered way

2.5.6. Abort

Abort


Abort

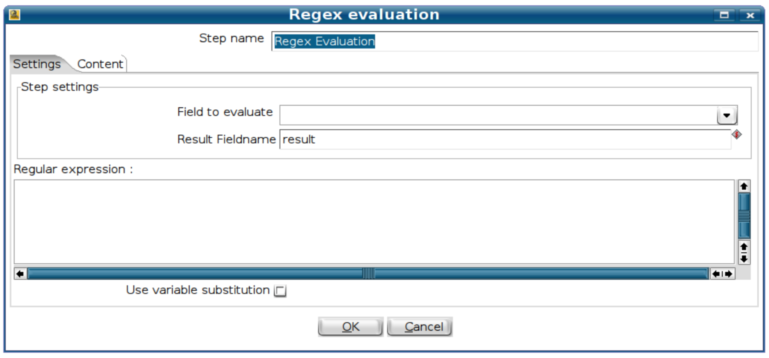
Step name: Abort
Abort threshold: 0
Abort message:
Always log rows: ☒

OK Cancel

Aborts a transformation in case one or more rows are read on input.

2.5.7. Regex Evaluation

Regex Evaluation


Regex evaluation

Step name: Regex Evaluation

Settings | Content

Step settings
Field to evaluate:
Result Fieldname: result

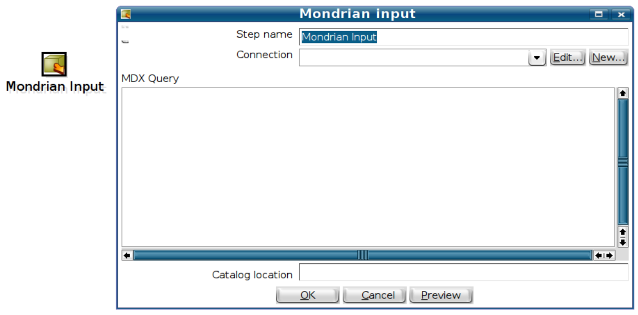
Regular expression :

Use variable substitution ☐

OK Cancel

Evaluate (and substitute) using regular expressions.

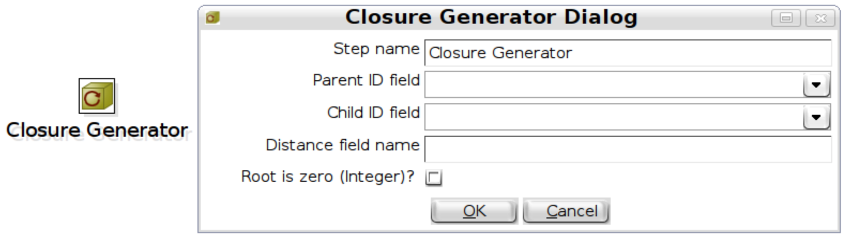
2.5.8. Mondrian Input



Step name: Mondrian Input

Read data from a Mondrian server using MDX.

2.5.9. Closure generator

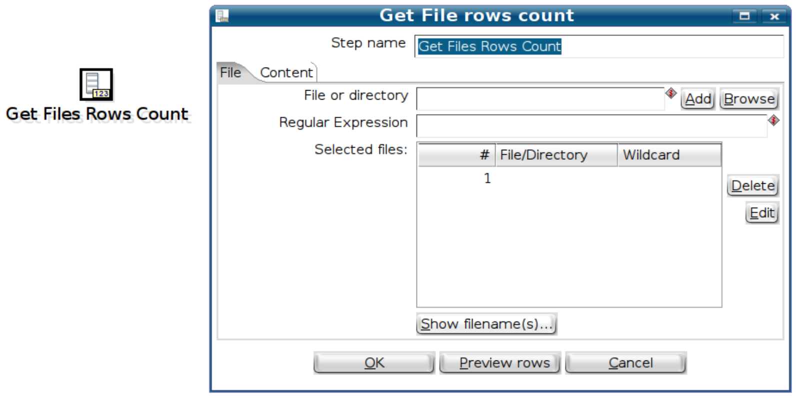


Step name: Closure Generator

Generates transitive closured for use with Mondrian.

See also: http://wiki.pentaho.org/display/EAI/Closure+Generator

2.5.10. Get files rows count



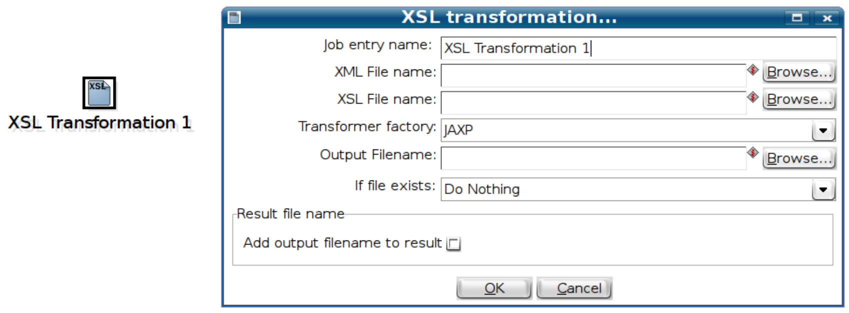
Step name: Get Files Rows Count

Count the number of rows in files.

2.6. New Job entries

Below is an overview of the new job entries that were added. For the full details on these job entries, please consult the Spoon user guide documentation.

2.6.1. XSL transformation



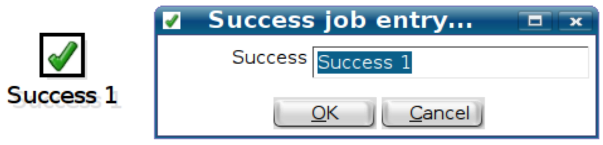
XSL Transformation 1

XSL transformation...

Job entry name: XSL Transformation 1
XML File name: Browse...
XSL File name: Browse...
Transformer factory: JAXP
Output Filename: Browse...
If file exists: Do Nothing
Result file name
Add output filename to result ☐

A job entry to transform XML documents using XSLT

2.6.2. Success



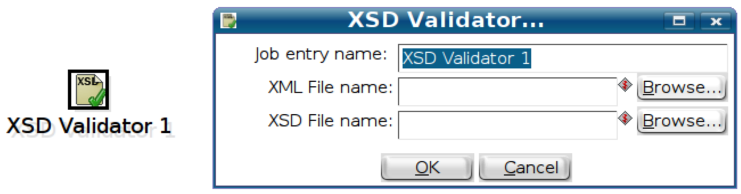
Success 1

Success job entry...

Success: Success 1

To force success on a job, for example after an expected failure

2.6.3. XSD validator



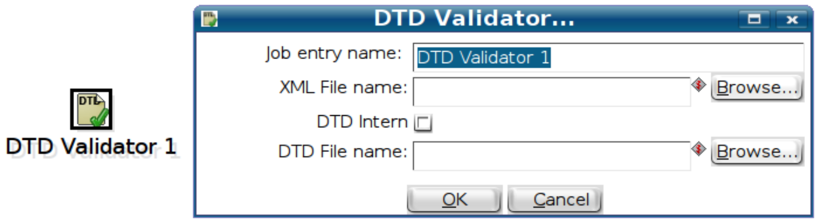
XSD Validator 1

XSD Validator...

Job entry name: XSD Validator 1
XML File name: Browse...
XSD File name: Browse...

Validate an XML document using XSD

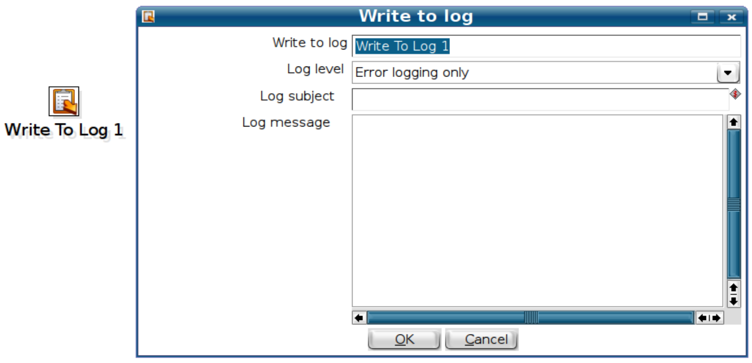
2.6.4. DTD validator



Job entry name: DTD Validator 1
XML File name:
DTD Intern
DTD File name:

Validate an XML document using DTD

2.6.5. Write to log file

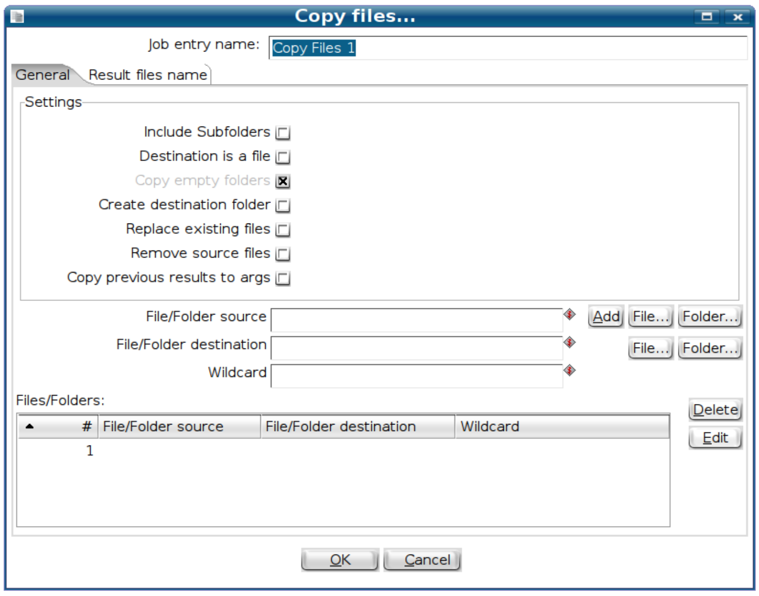


Write to log: Write To Log 1
Log level: Error logging only
Log subject:
Log message:

Write a message in the log file

2.6.6. Copy files

Copy Files 1



Job entry name: Copy Files 1

Include Subfolders ☐
Destination is a file ☐
Copy empty folders ☒
Create destination folder ☐
Replace existing files ☐
Remove source files ☐
Copy previous results to args ☐

File/Folder source
File/Folder destination
Wildcard

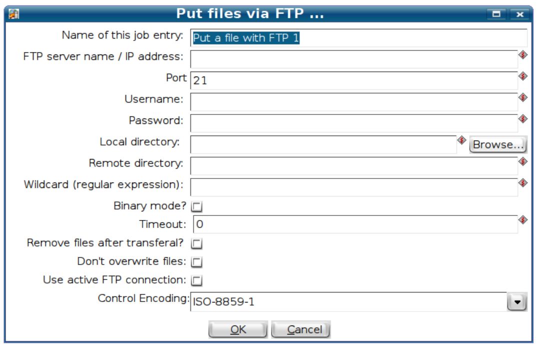
Files/Folders:

#	File/Folder source	File/Folder destination	Wildcard
1			

Copies one or more sets of files from one location to another

2.6.7. Put a file with FTP

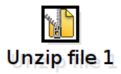
Put a file with FTP 1

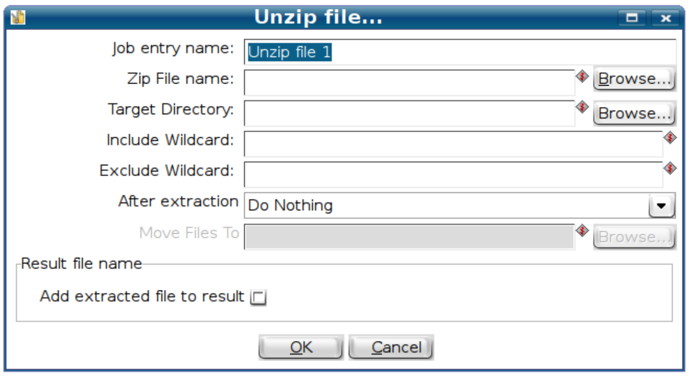


Name of this job entry:	Put a file with FTP 1
FTP server name / IP address:	
Port	21
Username:	
Password:	
Local directory:	
Remote directory:	
Wildcard (regular expression):	
Binary mode?	☐
Timeout:	0
Remove files after transferal?	☐
Don't overwrite files:	☐
Use active FTP connection:	☐
Control Encoding:	ISO-8859-1

Allows you to transfer multiple local files to a remove FTP server.

2.6.8. Unzip files




Unzip file...

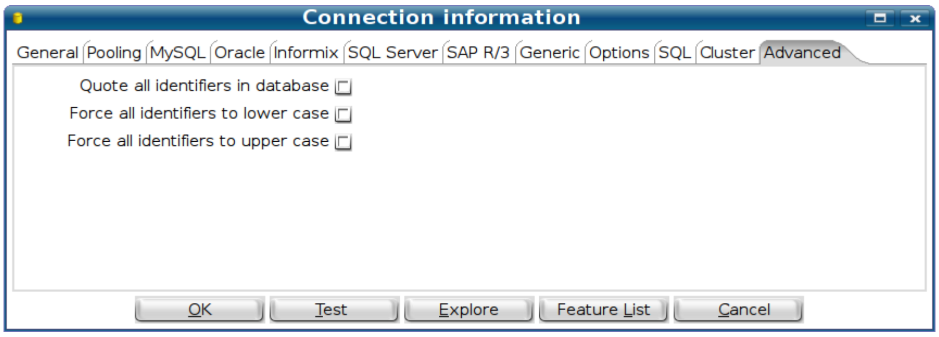
Job entry name: Unzip file 1
Zip File name:
Target Directory:
Include Wildcard:
Exclude Wildcard:
After extraction: Do Nothing
Move Files To

Result file name

Add extracted file to result ☐

2.7. Databases

- We now also have a set of advanced options to steer the behavior of table and column (field) quoting:



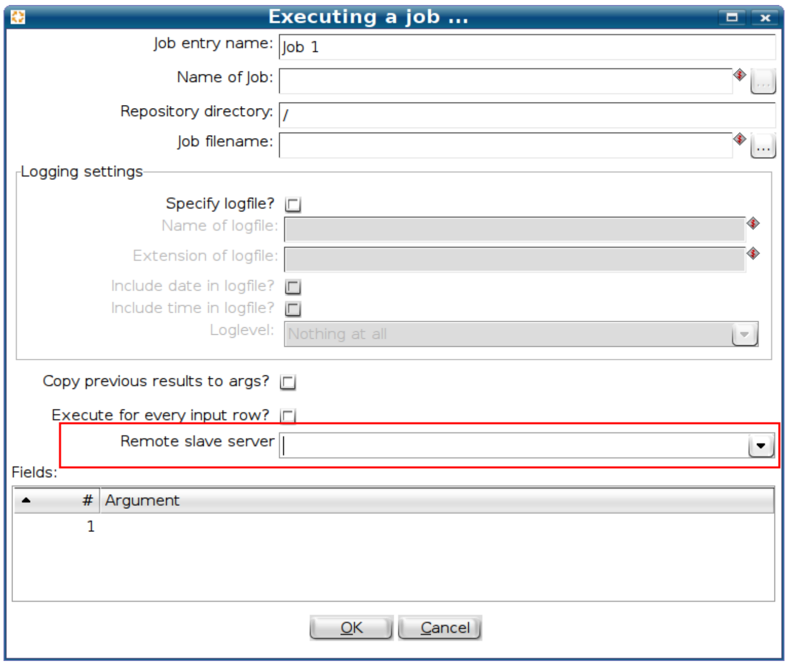
Quote all identifiers in database ☐
Force all identifiers to lower case ☐
Force all identifiers to upper case ☐

- Support was added for Apache Derby, BMC Remedy Action Request System and Sybase IQ.
- We added a PALO database placeholder for a set of plugins that will arrive soon. This is very much like we did for SAP/R3.

2.8. Slave server improvements

2.8.1. Remote execution of jobs

It is now possible to execute complete jobs to a remote slave server. This is not only possible from the Spoon job execution configuration dialog, but also from within a job:



Job entry name: Job 1
Name of Job:
Repository directory: /
Job filename:
Specify logfile? ☐
Name of logfile:
Extension of logfile:
Include date in logfile? ☐
Include time in logfile? ☐
Loglevel: Nothing at all
Copy previous results to args? ☐
Execute for every input row? ☐
Remote slave server

#	Argument
1	

2.8.2. Improved partitioning support

It is now possible to specify how many partitions should be created on each slave server in case you are running in a cluster. (A step running on a cluster schema)

It is now simply possible to specify the number of partitions to create per slave server. For example if we have a step called “A” running on 6 slave servers in a cluster schema and that step is partitioned to run with 9 copies per slave server, we will run 54 copies of the same step in parallel on 6 hosts in 54 threads. Each step will receive data that is meant for that specific partition.

We also allow you to re-partition data from one step to the next in a cluster. For example if you have a step “B” after step “A” running on the same cluster with a different partition schema that only starts 3 steps per slave we will re-partition the data from 54 steps to the 18 target steps. Since all possible copies of step “A” need to be able to write to all possible copies of step “B”, we create 972 communications channels over TCP/IP ports to make this happen.

Additionally, to allow companies to use their own partitioning logic, we allow you to create and install partitioning plugins.

3. Source code improvements

3.1. A few extra lines of code

Version 2.1.4 contains 160,000 lines of code.

Version 2.2.2 contains 177,450 lines of code, an increase of 17,450 lines.

Version 2.3.0 contains 213,489 lines of code, an increase of 36,039 lines.

Version 2.4.0 contains 256,030 lines of code, an increase of 42,541 lines.

Version 2.5.0 contains 292,241 lines of code, an increase of 36,211 lines.

Version 3.0.0 contains 348,575 lines of code, an increase of 56,334 lines.

3.2. Core committers

Here is the list of all the people that helped to make version 3.0 happen. (not in any particular order!)

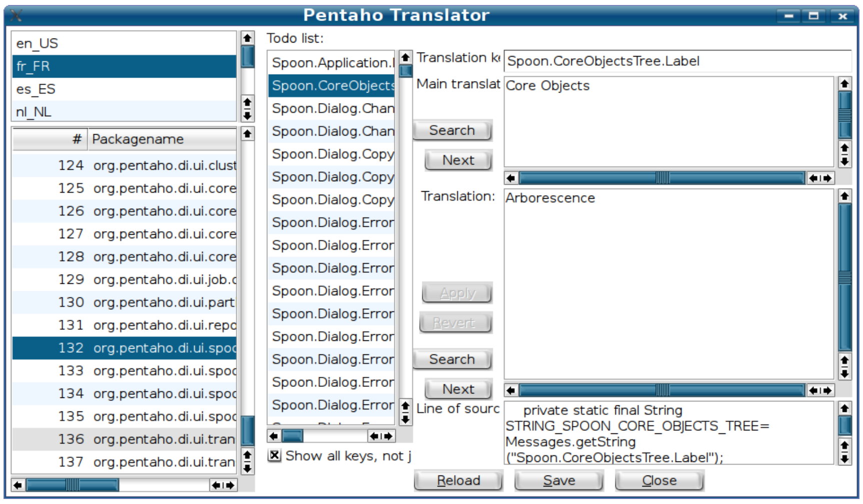
ID	e-Mail	Name	Country
sboden	Svenboden (at) hotmail.com	Sven Boden	B
jbleuel	Jens (at) pentaho.org	Jens Bleuel	D
Sven.thiergen	s.thiergen (a) itcampus.de	Sven Thiergen	D
shassan2	Sahass78 (a) yahoo.fr	Samatar Hassan	F
molm	manfred.olm (a) free.fr	Manfred Sherlock Olm	F
hdupre	henri.dupre (a) gmail.com	Henri Dupre	F
asilva	asilva (a) pentaho.org	Alex Silva	US
jdixon	jdixon (a) pentaho.org	James Dixon	US
Brassrat	jgoldman (a) vestmark.com	Jay Goldman	US
mbatchelor	mbatchelor (a) pentaho.org	Marc Matchelor	US
mlowerly	mlowery (a) pentaho.org	Matthew Lowery	US
quinhui99	qinhui99 (a) hotmail.com	Qin Hui (Tom Qin)	CH
wgorman	wgorman (a) pentaho.org	Will Gorman	US
dkincade	dkinkade (a) pentaho.org	David Kincade	US
mdamour	mdamour (a) pentaho.org	Mike D'Amour	US
gmoran	gmoran (a) pentaho.org	Gretchen Moran	US
bhagan	bhagan (a) pentaho.org	Brian Hagan	US
jtcornelius	jtcornelius (a) pentaho.org	Jake Cornelius	US
arodriguez	arodriguez (a) pentaho.org	Anthony Rodriguez	US
sbarkdull	sbarkdull (a) pentaho.org	Steven Barkdull	US
dmoran	dmoran (a) pentaho.org	Doug Moran	US
mcasters	mcasters (at) pentaho.org	Matt Casters	B

Because of the massive amount of commits (+2000) and as usual space and time constraints, it's very difficult to include a detailed commit log. It's also very hard to quantify the amount of work someone did as it is very clear that we appreciate all the help we get on this project.

3.3. Translators

A special mention goes to Samatar Hassan for translating all 6000 keys into fr_FR (French from France).

A lot of action is going on at the moment of writing with translations of Kettle being lined up. The new translator GUI is probably at the root of this translation frenzy. To all people involved: a big thank you!!



en_US
fr_FR
es_ES
nl_NL

#	Packagename
124	org.pentaho.di.ui.clust
125	org.pentaho.di.ui.core
126	org.pentaho.di.ui.core
127	org.pentaho.di.ui.core
128	org.pentaho.di.ui.core
129	org.pentaho.di.ui.job.c
130	org.pentaho.di.ui.part
131	org.pentaho.di.ui.repo
132	org.pentaho.di.ui.spoo
133	org.pentaho.di.ui.spoo
134	org.pentaho.di.ui.spoo
135	org.pentaho.di.ui.spoo
136	org.pentaho.di.ui.tran
137	org.pentaho.di.ui.tran

Spoon.Application.I
Spoon.CoreObjects
Spoon.Dialog.Chan
Spoon.Dialog.Chan
Spoon.Dialog.Copy
Spoon.Dialog.Copy
Spoon.Dialog.Copy
Spoon.Dialog.Error
Spoon.Dialog.Error
Spoon.Dialog.Error
Spoon.Dialog.Error
Spoon.Dialog.Error
Spoon.Dialog.Error
Spoon.Dialog.Error
Spoon.Dialog.Error
Spoon.Dialog.Error

Translation key: Spoon.CoreObjectsTree.Label

Main translation: Core Objects

Translation: Arborescence

```
    private static final String
STRING_SPOON_CORE_OBJECTS_TREE=
Messages.getString
("Spoon.CoreObjectsTree.Label");
```

3.4. Commit stats

In total we had in excess of 2000 revisions of the source code (commits).

All 1154 files in the subversion repository have been altered or re-created since we moved to a different package naming convention for Pentaho Data Integration. There isn't a single source code file that was not changed.

Bug reporters

Thank you all! Without you, it would be impossible to get PDI as stable as it is today.

3.5. Feature requesters

Thank you all for the many good suggestions! Sometimes the simple things have a great impact on the software. Good ideas really make the difference.