


User manual Last updated: 12-07-2006	Pentaho Data Integration	 pentaho™ open source business intelligence™
	Spoon 2.3.0	



Spoon version 2.3.0

© 2001 - 2006 Pentaho

*<http://www.pentaho.org>*

## Table of contents

1. About this document.....	5
1.1. What it is.....	5
1.2. What it is not.....	5
1.3. Other documentation.....	5
2. Spoon.....	6
2.1. What is spoon?.....	6
2.2. Installation.....	6
2.3. Launching spoon.....	6
2.4. Supported platforms.....	7
2.5. Known issues.....	7
2.6. Screen shot.....	8
2.7. Command line options.....	8
2.8. Repository.....	9
2.9. Repository Auto-Login.....	10
2.10. License.....	10
2.11. Notes.....	10
2.12. Definitions.....	10
2.13. Toolbar.....	12
2.14. Options.....	13
2.14.1. General tab.....	14
2.14.2. Look & Feel tab.....	15
2.15. Search meta-data.....	16
2.16. Set environment variable.....	16
2.17. Execution log history.....	16
2.18. Replay.....	17
2.19. Generate mapping against target step.....	17
2.20. Safe mode.....	18
3. Database Connections.....	19
3.1. Description.....	19
3.2. Screen shot.....	19
3.3. Options.....	19
3.4. Database Usage Grid.....	20
3.5. Usage.....	21
3.5.1. Create a new connection.....	21
3.5.2. Edit a connection.....	21
3.5.3. Duplicate a connection.....	21
3.5.4. Copy to clipboard.....	21
3.5.5. Delete a connection.....	21
3.5.6. Test a connection.....	22
3.5.7. Execute SQL commands on a connection.....	22
3.5.8. Clear DB Cache option.....	22
3.5.9. Explore.....	22
3.6. Unsupported databases.....	22
4. SQL editor.....	23
4.1. Description.....	23
4.2. Screen shot.....	23
5. Database Explorer.....	24
5.1. Screen shot .....	24
5.2. Description.....	24
6. Hops.....	25
6.1. Description.....	25
6.2. Screen shot.....	25
6.3. Creating A Hop.....	25

6.4. Splitting A Hop.....	25
6.5. Loops.....	25
7. Transformation settings.....	26
7.1. Description.....	26
7.2. Screen shots.....	26
7.3. Options.....	28
7.4. Extra.....	29
8. Steps.....	30
8.1. Description.....	30
8.2. Launching several copies of a step.....	30
8.3. Distribute or copy?.....	32
8.4. Use of variables.....	33
8.5. Step Types.....	34
8.5.1. Text File Input .....	34
8.5.2. Text File Output.....	44
8.5.3. Table input.....	47
8.5.4. Table output.....	50
8.5.5. Select values.....	52
8.5.6. Filter rows.....	54
8.5.7. Database lookup.....	56
8.5.8. Sort rows.....	57
8.5.9. Stream lookup.....	58
8.5.10. Add sequence.....	60
8.5.11. Dimension lookup/update.....	62
8.5.12. Combination lookup/update.....	65
8.5.13. Dummy (do nothing).....	67
8.5.14. Join Rows (Cartesian product).....	69
8.5.15. Aggregate Rows.....	71
8.5.16. Get System Info.....	72
8.5.17. Generate Rows.....	74
8.5.18. Java Script Value.....	75
8.5.19. Call DB Procedure.....	85
8.5.20. Insert / Update.....	86
8.5.21. Update.....	87
8.5.22. Row Normaliser.....	88
8.5.23. Denormaliser.....	91
8.5.24. Flattener.....	92
8.5.25. Split Fields.....	94
8.5.26. Unique rows.....	96
8.5.27. Group By.....	97
8.5.28. Copy rows to result.....	98
8.5.29. Get rows from result.....	98
8.5.30. XBase input.....	99
8.5.31. Excel input.....	100
8.5.32. Database Join.....	102
8.5.33. Cube output.....	103
8.5.34. Cube input.....	103
8.5.35. Calculator.....	104
8.5.36. Merge rows.....	107
8.5.37. Add constants.....	108
8.5.38. Execute SQL script.....	109
8.5.39. Mapping.....	111
8.5.40. XML Input.....	114
8.5.41. XML Output.....	117
8.5.42. Delete.....	119

8.5.43. Value Mapper.....	120
8.5.44. Set Variable.....	121
8.5.45. Get Variable.....	122
8.5.46. Get Filename.....	123
8.5.47. Get files from result.....	124
8.5.48. Set files in result.....	125
8.5.49. Blocking step.....	126
8.5.50. Injector.....	127
8.5.51. Null If.....	128
9. Graphical View.....	129
9.1. Description.....	129
9.2. Adding steps.....	129
9.2.1. Create steps by drag and drop.....	129
9.2.2. Create steps from the step types tree.....	130
9.2.3. Create steps on the place where you like them.....	131
9.3. Hiding a step.....	131
9.4. Step options (popup menu).....	131
9.5. Edit step.....	131
9.6. Edit description.....	131
9.7. Copy data / distribute data.....	131
9.8. Copies.....	131
9.9. Duplicate step .....	131
9.10. Delete step.....	131
9.11. Show input fields.....	131
9.12. Show output fields.....	132
9.13. Adding hops.....	132
10. Log View.....	133
10.1. Description.....	133
10.2. Screenshot.....	133
10.3. Log Grid.....	134
10.4. Buttons.....	134
10.4.1. Start transformation.....	134
10.4.2. Preview.....	135
10.4.3. Show error lines.....	135
10.4.4. Clear log.....	135
10.4.5. Log Settings.....	135
10.4.6. Hide inactive.....	136
10.4.7. Safe mode.....	136
11. Grids.....	137
11.1. Description.....	137
11.2. Functions.....	137
11.3. Navigating.....	137
12. Repository Explorer.....	138
12.1. Description.....	138
12.2. Screenshot.....	138
12.3. Functions.....	138
12.4. Backup / Recovery.....	139
13. Apendix A.....	140

## 1. ABOUT THIS DOCUMENT

---

### 1.1. What it is

This document is a technical description of Spoon, the graphical transformation designer of the Pentaho Data Integration suite a.k.a. Kettle.

### 1.2. What it is not

In this document we do not describe in great detail on **how** to create transformations in all possible situations. Obviously, different developers might take a different route to creating a data integration solution. And that's the way it should be. Spoon is trying to give the users of this tool a freedom in how to implement solutions.

### 1.3. Other documentation

Here are links to other documents that you might be interesting to go through when you're building transformations:

Flash demos and an introduction to building a simple transformation:

<http://www.kettle.be/en/screenshots.htm>

The Kettle community website (Tips are available under Documents / Documentation):

<http://kettle.javaforge.com>

Running transformations in batch using Pan: [Pan-2.3.pdf](#)  
Chef, the graphical Job designer: [Chef-2.3.pdf](#)  
Running jobs in batch using Kitchen: [Kitchen-2.3.pdf](#)

The Kettle community website (Tips are available under Documents / Documentation):

<http://kettle.javaforge.com>

The Pentaho website containing technical tips, forum, etc: <http://www.pentaho.org>

An introduction to Pentaho Data Integration in Roland Bouman's blog:

<http://rpbouman.blogspot.com/2006/06/pentaho-data-integration-kettle-turns.html>

Nicholas Goodman is also blogging on Kettle and BI: <http://www.nicholasgoodman.com>  
(*Nick is director of Business Intelligence at Pentaho*)

## 2. SPOON

### 2.1. What is spoon?

Kettle is an acronym for “Kettle E.T.T.L. Environment”. This means it has been designed to help you with your ETLT needs: the Extraction, Transformation, Transportation and Loading of data.

Spoon is a graphical user interface that allows you to design transformations that can be run with the Kettle tool Pan. Pan is a data transformation engine that is capable of performing a multitude of functions such as reading, manipulating and writing data to and from various data sources.

**NOTE:** *For a complete description of Pan, please check out the Pan documentation.*

Transformations can describe themselves using an XML file or can be put in a Kettle database repository. This information can then be read by Pan to execute the described steps in the transformation.

In short: ***Kettle makes data warehouses easier to build, update and maintain!***

### 2.2. Installation

The first step is the installation of Sun Microsystems Java Runtime Environment version 1.4 or higher. You can download a JRE for free at <http://www.javasoft.com/>.

After this, you can simply unzip the zip-file: Kettle-2.3.0.zip in a directory of your choice. In the Kettle directory where you unzipped the file, you will find a number of files. Under Unix-like environments (Solaris, Linux, MacOS, ...) you will need to make the shell scripts executable. Execute these commands to make all shell scripts in the Kettle directory executable:

```
cd Kettle
chmod +x *.sh
```

### 2.3. Launching spoon

To launch Spoon on the different platforms these are the scripts that are provided:

- ✓ Spoon.bat: launch Spoon on the Windows platform.
- ✓ spoon.sh: launch Spoon on a Unix-like platform: Linux, Apple OSX, Solaris, ...

If you want to make a shortcut under the Windows platform an icon is provided: “spoon.ico” to set the correct icon. Simply point the shortcut to the Spoon.bat file.

## 2.4. Supported platforms

The Spoon GUI is supported on the following platforms:

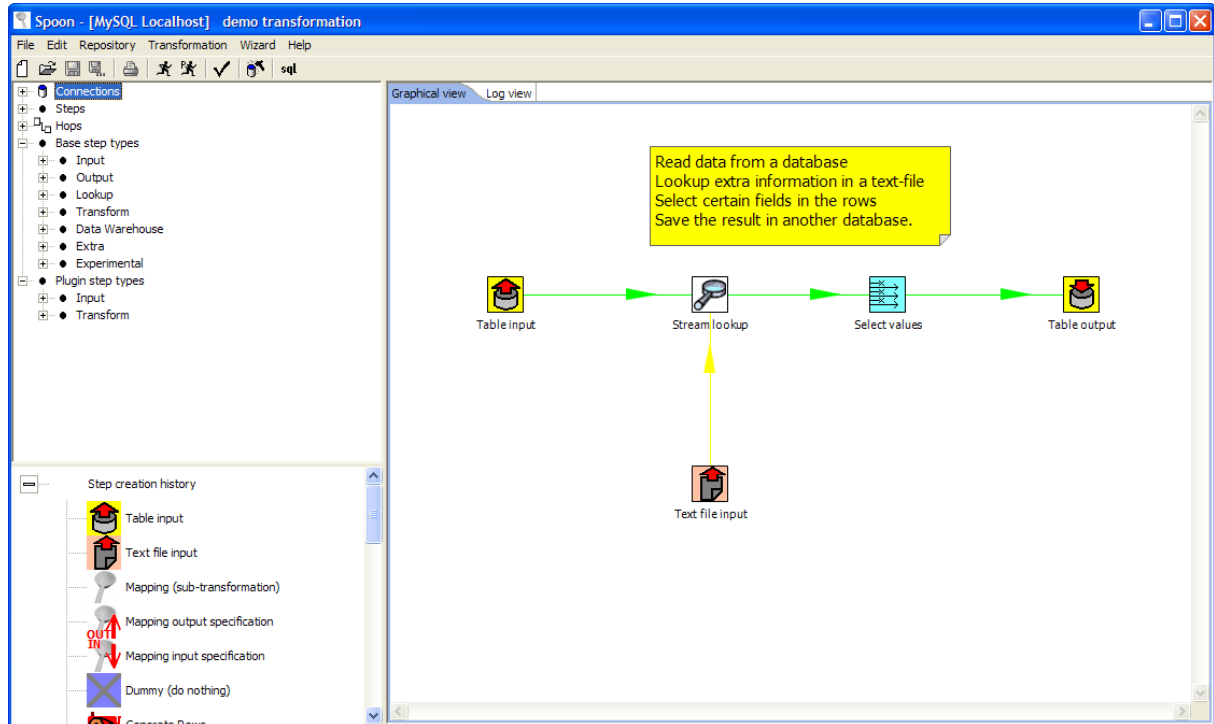
- Microsoft Windows: all platforms since Windows 95, including Vista
- Linux GTK: on i386 and x86\_64 processors, works best on Gnome
- Apple's OSX: works both on PowerPC and Intel machines
- Solaris: using a Motif interface (GTK optional)
- AIX: using a Motif interface
- HP-UX: using a Motif interface (GTK optional)
- FreeBSD: preliminary support on i386, not yet on x86\_64

## 2.5. Known issues

- Linux
  - Occasional JVM crashes running SuSE Linux and KDE. Running under Gnome has no problems. (detected on SUSE Linux 10.1 but earlier versions suffer the same problem)
- FreeBSD
  - Problems with drag and drop. Workaround is to use the right click popup menu on the canvas. (Insert new step)

Please check the Tracker lists at <http://kettle.javaforge.com> for up-to-date information on discovered issues.

## 2.6. Screen shot



The following items are visible in Spoon: Connections, Steps, Hops, Base Step Types, Graphical View and the Log View.

These items are described in detailed in the chapters below: Connections in [2. Database Connections](#), Hops in [5. Hops](#), Steps in [7. Steps](#), the Graphical View in [8. Graphical View](#).

## 2.7. Command line options

These are the command line options that you can use.

**NOTE:** On Windows we advice you to use the */option:value* format to avoid command line parsing problems by the MS-DOS shell.

Fields in *italic* represent the values that the options use.

```
-file=filename
```

This option runs the transformation defined in the XML file. (.ktr : Kettle Transformation)

```
-logfile=Logging Filename
```

Specifies the log file. The default is the standard output.


```
-level=Logging Level
```

The level option sets the log level for the transformation that's being run.

These are the possible values:

- ✓ Nothing: Don't show any output
- ✓ Error: Only show errors



User manual Last updated: 12-07-2006	Pentaho Data Integration	
	Spoon 2.3.0	

- ✓ Minimal: Only use minimal logging
- ✓ Basic: This is the default basic logging level
- ✓ Detailed: Give detailed logging output
- ✓ Debug: For debugging purposes, very detailed output.
- ✓ Rowlevel: Logging at a row level, this can generate a lot of data.

```
-rep=Repository name
```

Connect to the repository with name “*Repository name*”.

You also need to specify the options `-user`, `-pass` and `-trans`.

The repository details are loaded from the file `repositories.xml` in the local directory or in the Kettle directory: `$HOME/.kettle/` or `C:\Documents and Settings\<username>\.kettle` on Windows.

```
-user=Username
```

This is the username with which you want to connect to the repository.

```
-pass=Password
```

The password to use to connect to the repository

```
-trans=Transformation Name
```

Use this option to select the transformation to run from the repository


**NOTE:** *It's important that if spaces are present in the option values, you use quotes or double quotes to keep them together.*

## 2.8. Repository

A Kettle repository can contain among other things transformations. This means that in order to load a transformation from a database repository, you need to connect to this repository.

To do this, you need to define a database connection to this repository. You can do this using the repositories dialog you are presented with when you start up Spoon.



User manual Last updated: 12-07-2006	Pentaho Data Integration	
	Spoon 2.3.0	

The information concerning repositories is stored in file called "repositories.xml". This file resides in the hidden directory ".kettle" in your default home directory. On windows this is C:\Documents and Settings\<username>\.kettle

**HINT:** *The complete path and filename of this file is displayed on the Spoon console.*

If you don't want this dialog to be shown each time Spoon starts up, you can disable it using the Options dialog under the Edit / Options menu. See also [1.11. Options](#)

**IMPORTANT:** *The default password for the `admin` user is also `admin`. You should change this default password right after the creation using the Repository Explorer.*

## 2.9. Repository Auto-Login

You can have Spoon automatically log into the repository by setting the following environment variables: KETTLE\_REPOSITORY, KETTLE\_USER and KETTLE\_PASSWORD.

This saves you from having to log into the same repository every time. ***Please note that this is a security risk and that you should always lock your computer to prevent unauthorized access to the repository.***

## 2.10. License

Starting with version 2.2.0, Kettle was released into the public domain under the LGPL license. Please refer to Appendix A for the full text of this license.

**Note:** *Pentaho Data Integration is referred to as "Kettle" below.*

Copyright (C) 2006 Pentaho Corporation

Kettle is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

Kettle is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the Kettle distribution; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

## 2.11. Notes

You can put notes on the graphical view everywhere simply by clicking right on the canvas and selecting "Add note".

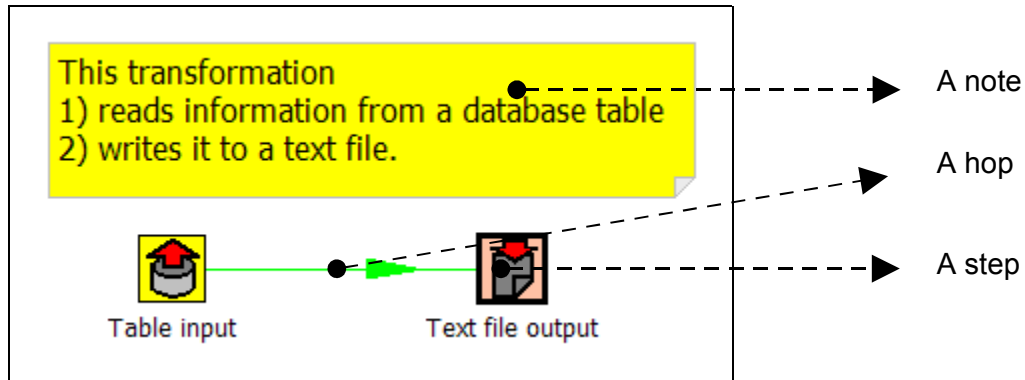
Later these notes can be edited by double clicking on them and dragged around the screen by dragging on them with the mouse using the left button.

Removing a note can be done by a right click on the note and by selecting "Delete note".

## 2.12. Definitions












- ✓ **Value:** Values are part of a row and can contain any type of data: [Strings](#), floating point [Numbers](#), unlimited precision [BigNumbers](#), [Integers](#), [Dates](#) or [Boolean](#) values.
- ✓ **Row:** a row exists of 0 or more values
- ✓ **Output stream:** an output stream is a stack of rows that leaves a step.
- ✓ **Input stream:** an input stream is a stack of rows that enters a step.

- ✓ **Hop:** a hop is a graphical representation of one or more data streams between 2 steps.  
 A hop always represents the output stream for one step and the input stream for another.  
 The number of streams is equal to the copies of the destination step. (1 or more)
- ✓ **Note:** a note is a piece of information that can be added to a transformation



## 2.13. Toolbar

The icons on the toolbar of the main screen are from left to right:

<i>Icon</i>	<i>Meaning</i>
	New transformation
	Open transformation from file if you're not connected to a repository or from the repository if you are connected to one.
	Save the transformation to a file or to the repository.
	Save the transformation under a different name or filename.
	Print: you will be presented with a print-dialog asking you to specify the number of pages, margins etc.
	Run transformation: runs the current transformation from XML file or repository.
	Preview transformation: runs the current transformation from memory. You can preview the rows that are produced by selected steps.
	Replay the processing of a transformation for a certain date and time. This will cause certain steps (Text File Input and Excel Input) to only process rows that failed to be interpreted correctly during the run on that particular date and time.
	Verify transformation: Spoon runs a number of checks for every step to see if everything is going to run as it should.
	Run an impact analysis: what impact does the transformation have on the used databases.
	Generate the SQL that is needed to run the loaded transformation.

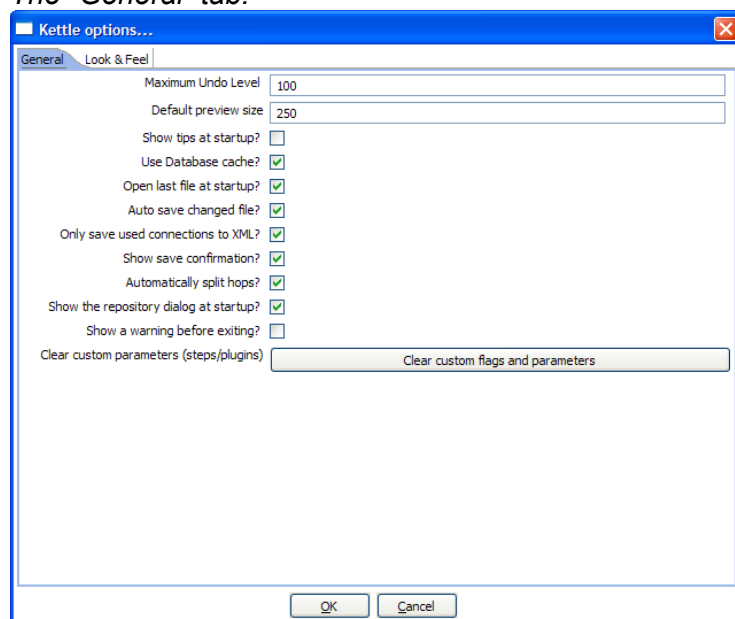
## 2.14. Options

There are a number of options that you can change to enhance the graphical user interface, such as the fonts and the colors of the screens.

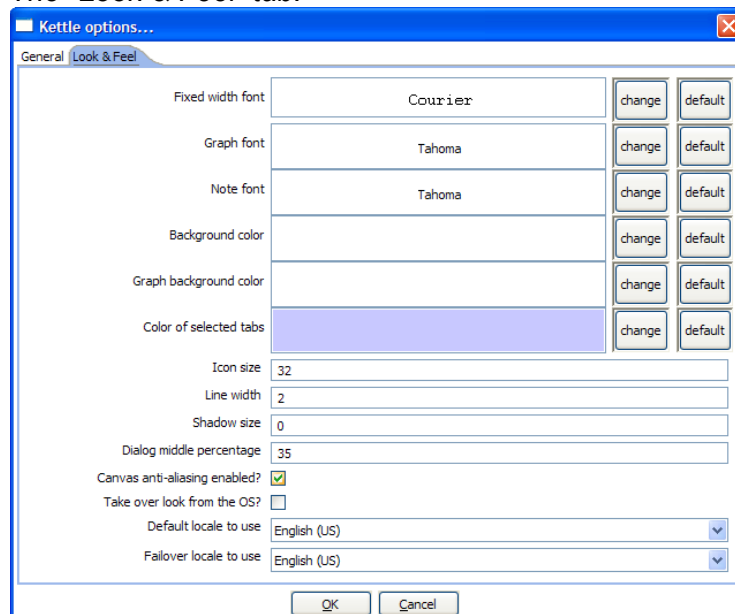
To change these options, please check out the Edit menu option “Options”.

These are screen shots of the options:

*The “General” tab:*



*The “Look & Feel” tab:*



## 2.14.1. General tab

### 2.14.1.1. Maximum Undo Level

This parameter sets the maximum number of steps that can be undone (or redone) by Spoon.

### 2.14.1.2. Default preview size

This parameter allows you to change the default number of rows that are requested from a step during transformation previews.

### 2.14.1.3. Show tips at startup

This option sets the display of tips at startup.

### 2.14.1.4. Use database cache

Spoon caches information that is stored on source and target databases. In some cases this can lead to incorrect results when you're in the process of changing those very databases. In those cases it is possible to disable the cache altogether instead of clearing the cache every time.

NOTE: Spoon automatically clears the database cache when you launch DDL (Data Definition Language) statements towards a database connection. However, when using 3<sup>rd</sup> party tools, clearing the database cache manually may be necessary.

### 2.14.1.5. Open last file at startup

Enable this option to automatically (try to) load the last transformation you used (opened or saved) from XML or repository.

### 2.14.1.6. auto save changed file

This option automatically saves a changed transformation before running.

### 2.14.1.7. Only save used connection in XML

This option limits the XML export of a transformation to the used connections in that transformation. This comes in handy while exchanging sample transformations to avoid having all defined connections to be included.

### 2.14.1.8. show save confirmation

This flag allows you to turn off the confirmation dialogs you receive when a transformation has been changed.

### 2.14.1.9. automatically split hops

This option turns off the confirmation dialogs you get when you want to split a hop. (see also [5.4. Splitting A Hop](#))

### 2.14.1.10. show the repositories dialog at startup

This option controls whether or not the repositories dialog shows up at startup.

### 2.14.1.11. Clear custom flags and parameters

This option clears all parameters and flags that were set in the plugin or step dialogs.

## 2.14.2. Look & Feel tab

### 2.14.2.1. Default font

This is the font that is used in the dialog boxes, trees, input fields, etc.

### 2.14.2.2. Graph font

This is the font that is used on the graphical view.

### 2.14.2.3. Grid font

This font is used in all the grids that are used in Spoon.

### 2.14.2.4. Note font

This font is used in the notes that are displayed in the Graphical View.

### 2.14.2.5. Background color

Sets the background color in Spoon. It affects all dialogs too.

### 2.14.2.6. Graph background color

Sets the background color in the Graphical View of Spoon.

### 2.14.2.7. Color of selected tabs

This is the color that is being used to indicate tabs that are active/selected.

### 2.14.2.8. Icon size

This affects the size of the icons in the graph window. The original size of an icon is 32x32 pixels. The best results (graphically) are probably at sizes 16,24,32,48,64 and other multiples of 32.

### 2.14.2.9. Line width

This affects the line width of the hops on the Graphical View and the border around the steps.

### 2.14.2.10. Shadow size

If this size is larger than 0, a shadow of the steps, hops and notes is drawn on the canvas, making it look like the transformation floats above the canvas.

### 2.14.2.11. Dialog middle percentage

By default, a parameter is drawn at 35% of the width of the dialog, counted from the left. You can change this with this parameter. Perhaps this can be useful in cases where you use unusually large fonts.

### 2.14.2.12. Canvas anti-aliasing enabled?

Some platforms like Windows, OSX and Linux support anti-aliasing through GDI, Carbon or Cairo. Check this to enable smoother lines and icons in your graph view.

#### 2.14.2.13. Take over look from the OS?

Checking this on Windows allows you to use the default system settings for fonts and colors in Spoon. On other platforms, this is always the case.

#### 2.14.2.14. Default locale to use

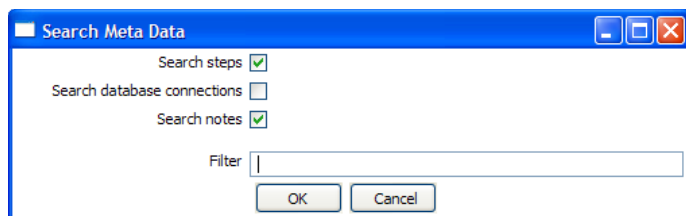
Here you can specify the default language setting. If a certain text hasn't been translated into this locale, Kettle will fall back to the fail over locale.

#### 2.14.2.15. Fail over locale to use

Because the original language in which Kettle was written is English, it's best to set this locale to English.

## 2.15. Search meta-data

This feature is accessible through the Edit menu or through CTRL-F.

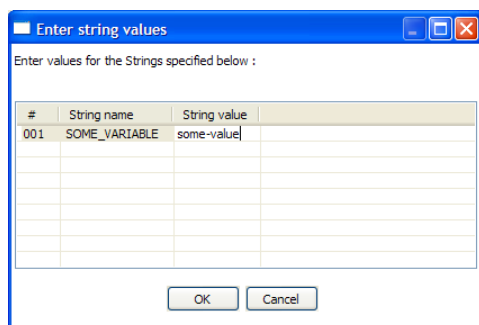


This option will search in the transformation in any possible field, connector or note.

A detailed result is shown.

## 2.16. Set environment variable

The possibility to set an environment variable was added because it becomes easier that way to test transformations that use dynamically set variables. Normally these variables are set by a different transformation in a job. However, during development and testing, you might want to set these manually.



This feature is available from the Edit menu as well as with the CTRL-J keyboard shortcut. This screen is also presented when you run a transformation that use undefined variables. That way you can define them right before execution time.

## 2.17. Execution log history

If you store the logging of the transformation in a database table (using the transformation settings, logging tab) you can see the overview of the runs in the log history tab in Spoon:



Graphical view														Log view	Log history
#	Batch ID	Status	Read	Written	Updated	Input	Output	Errors	Start date	End date	Log date	Dependency date	Replay date		
001	2	end	0	452	0	452	0	0	1900/01/01 00:00:00.000	2006/06/06 15:52:37.000	2006/06/06 15:52:37.000	2006/06/06 15:52:37.000	2006/06/06 15:52:37.000		
002	1	stop	0	308999	0	309001	0	0	1900/01/01 00:00:00.000	2006/06/06 15:51:41.000	2006/06/06 15:52:32.000	2006/06/06 15:51:41.000	2006/06/06 15:51:41.000		

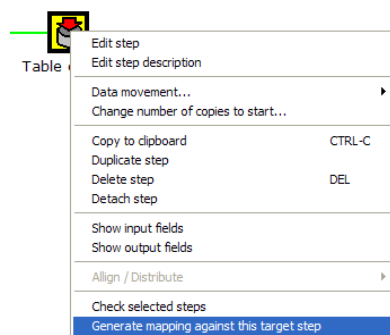
## 2.18. Replay

It is now possible to re-run a transformation that failed. Replay functionality is implemented for Text File Input and Excel input. It allows you to send files that had errors back to the source and have the data corrected. ONLY the lines that failed before are then processed during the replay if a .line file is present. It uses the date in the filename of the .line file to match the entered replay date.

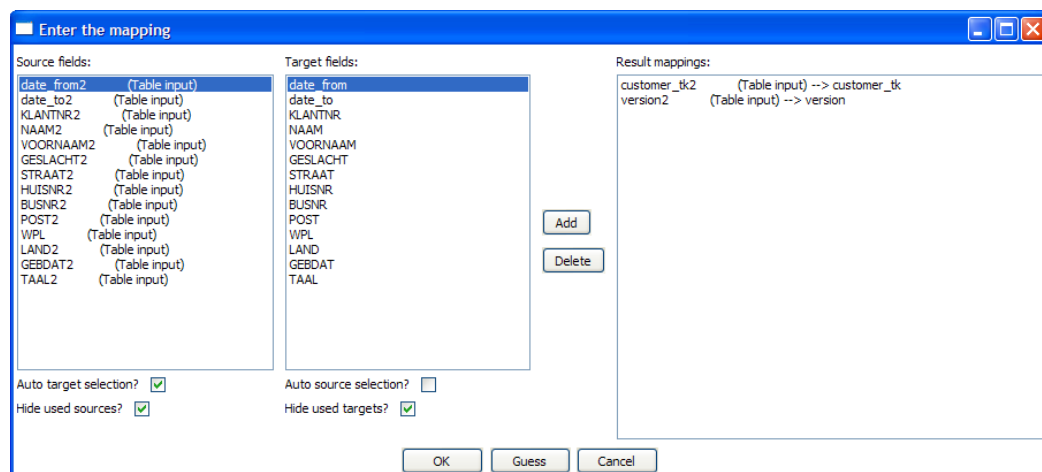


## 2.19. Generate mapping against target step

In cases where you have a fixed target table, you want to map the fields that go into the table output:

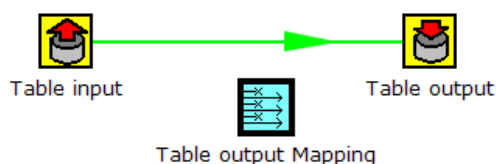


A dialog is then shown that helps you to determine which input fields goes to which table field:

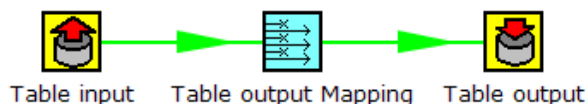


The selection of the target fields are done semi-automatic based on (parts of) the field name.

As a result of this dialog, a new Select Values step is generated:



We can simply place this step between the two and the mapping from input to output is done:



## 2.20. Safe mode

In cases where you are mixing the rows from various sources, you need to make sure that these row all have the same layout in all conditions. For this purpose, we added a “safe mode” option that is available in the Spoon logging window. When running in “safe mode”, the transformation will check every row that passes and will see if the layouts are all identical.

If a row is found that does not have the same layout as the first row, an error is thrown and the step and offending row are reported on.

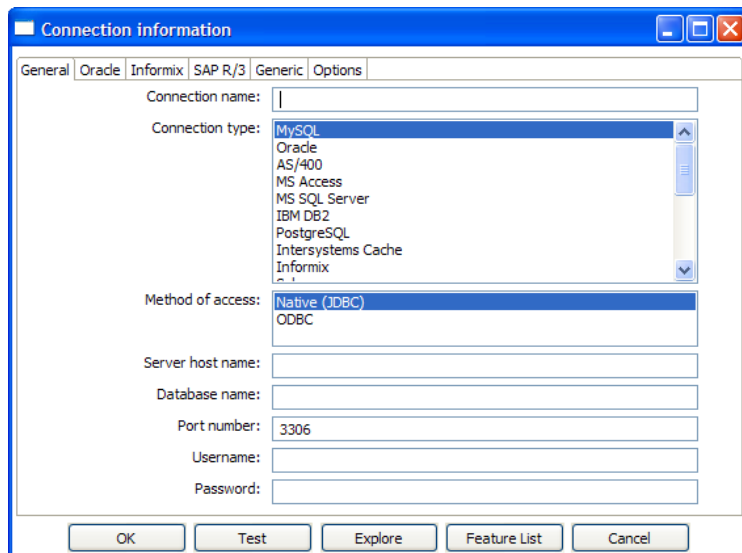
**Note:** this option is also available in Pan with the safemode option.

## 3. DATABASE CONNECTIONS

### 3.1. Description

A connection describes the method by which Kettle can connect to a database. The top entries in the tree on the left describe the available connections.

### 3.2. Screen shot



### 3.3. Options

- ✓ **Connection name:** a connection name uniquely name defines a connection across transformations.
- ✓ **Connection type:** the type of database you're connecting to.
- ✓ **Method of access:** This can be either Native (JDBC), ODBC, or OCI.
- ✓ **Server host name:** specify the host name of the server on which the database resides. You can also specify it's IP-address.
- ✓ **Database name:** identifies the database name you want to connect to. In case of ODBC specify the DSN name here (see also [2.4. Database Usage Grid](#)).
- ✓ **Port number:** sets the TCP/IP port number on which the database listens.
- ✓ **User name/password:** optionally specifies the user name and password to connect to the database.

#### EXTRA:

- For Oracle you can specify the default tablespaces in which Kettle will places objects generating SQL for tables and indexes.
- For Informix, you need to specify the Informix Server name in the Informix tab in order for a connection to be usable.
- For SAP R/3 connections, extra parameters Language, System Number and SAP Client can be specified in the SAP R/3 tab.
- Feature list: exposes the JDBC URL, class and various database settings for the connection such as the list of reserved words.
- Options: this new tab allows you to set database specific option on the database connections by adding parameters to the generated URL.

### 3.4. Database Usage Grid

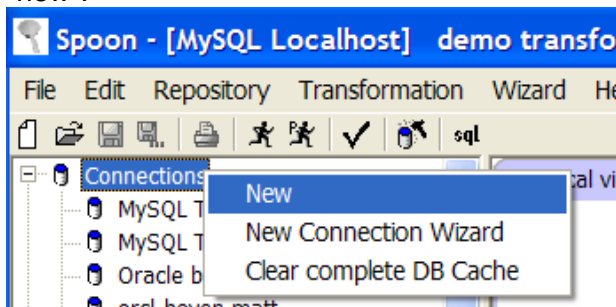
Database	Access Method	Server name or address	Database name	Port nr (default)	Username & password
Oracle	Native	Required	Oracle database SID	Required (1521)	Required
	ODBC		ODBC DSN name		Required
	OCI		Database TNS name		Required
MySQL	Native	Required	MySQL database name	Required (3306)	Required
	ODBC		ODBC DSN name		Required
AS/400	Native	Required	AS/400 Library name	Optional	Required
	ODBC		ODBC DSN name		Required
MS Access	ODBC		ODBC DSN name		Optional
MS SQL Server	Native	Required	Database name	Required (1433)	Required
	ODBC		ODBC DSN name		Required
IBM DB2	Native	Required	Database name	Required (50000)	Required
	ODBC		ODBC DSN name		Required
PostgreSQL	Native	Required	Database name	Required (5432)	Required
	ODBC		ODBC DSN name		Required
Intersystems Caché	Native	Required	Database name	Required (1972)	Required
	ODBC		ODBC DSN name		Required
Informix	Native	Required	Database name	Required (1526)	Required
	ODBC		ODBC DSN name		Required
Sybase	Native	Required	Database name	Required (5001)	Required
	ODBC		ODBC DSN name		Required
Gupta SQL Base	Native	Required	Database name	Required (2155)	Required
	ODBC		ODBC DSN name		Required
Dbase III, IV or 5.0	ODBC		ODBC DSN name		Optional
Firebird SQL	Native	Required	Database name	Required (3050)	Required
	ODBC		ODBC DSN name		Required
Hypersonic	Native	Required	Database name	Required (9001)	Required
MaxDB (SAP DB)	Native	Required	Database name		Required
	ODBC		ODBC DSN name		Required
CA Ingres	Native	Required	Database name		Required
	ODBC		ODBC DSN name		Required
Borland Interbase	Native	Required	Database name	Required (3050)	Required
	ODBC		ODBC DSN name		Required
ExtenDB	Native	Required	Database name	Required (6453)	Required
	ODBC		ODBC DSN name		Required
Generic <sup>(*)</sup>	Native	Required	Database name	Required (Any)	Required
	ODBC		ODBC DSN name		Optional

<sup>(\*)</sup>The generic database connection also needs to specify the URL and Driver class in the Generic tab!!

## 3.5. Usage

### 3.5.1. Create a new connection

You can create a new connection by right clicking on the "Connections" tree entry and selecting "new".

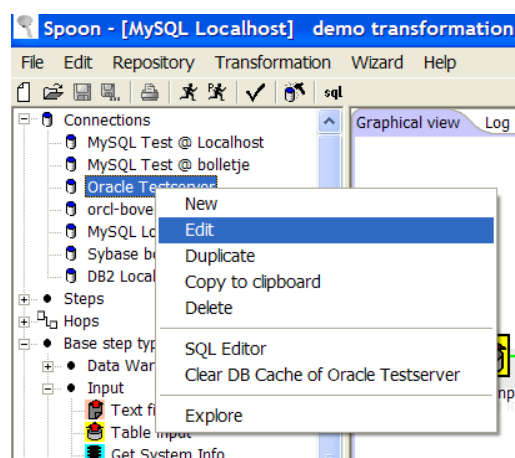


**Or** : simply double click on the Connections tree entry.

**Or** : press F3 to start the "new connection" wizard.

### 3.5.2. Edit a connection

Double click on the connection name in the left tree. Or right click on the name and select "Edit connection".



### 3.5.3. Duplicate a connection

Right click on the connection name and select "Duplicate".

### 3.5.4. Copy to clipboard

Copies the XML describing the connection to the clipboard.

### 3.5.5. Delete a connection

Right click on the connection name and select "Delete".

### 3.5.6. Test a connection

In the edit window (see above), select the "Test" button. If Spoon can connect, an OK message is displayed after a short delay.

### 3.5.7. Execute SQL commands on a connection

Right click on the connection name and select "SQL Editor". See also [3. SQL editor](#) for more information.

### 3.5.8. Clear DB Cache option

To speed up connections Spoon uses a database cache. Use this option when the information in the cache no longer represents the layout of the database. This is the case when databases tables have been changed, created or deleted.

### 3.5.9. Explore

This option will start the database explorer for the selected database connection. Please see [4. Database Explorer](#) for more information.

## 3.6. Unsupported databases

If you want to access a database type that is not yet supported, let us know and we will try to find a solution. A few database types are not supported in this release because of the lack of sample database and/or software.

Please note that it is usually still possible to read from these databases by using the Generic database driver through an ODBC connection.

## 4. SQL EDITOR

---

### 4.1. Description

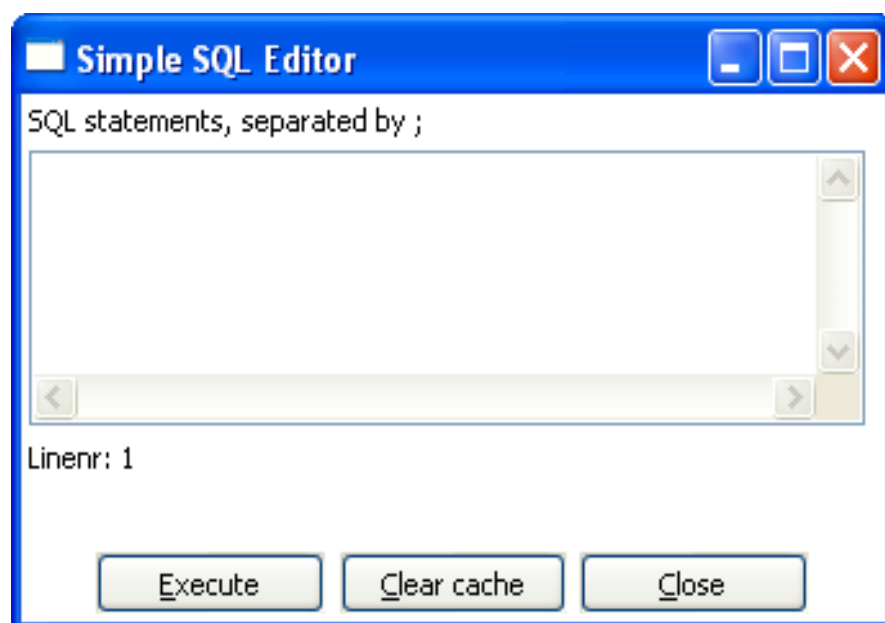
Sometimes a simple SQL Editor can be nice to have. Especially when you're creating tables, dropping indexes and modifying fields. The simple SQL editor supplied in Spoon, allows you to do this. In fact, most of the DDL (Data Definition Language) such as "create/alter table", "create index" and "create sequence" SQL commands are created automatically for you via the SQL Editor window.

**NOTE:** Multiple SQL Statements have to be separated by semi-colons (;).

**NOTE:** Before these SQL Statements are sent to the database to be executed, Spoon removes returns, line-feeds and the separating semi-colons.

**NOTE:** Kettle clears the database cache for the database connection on which you launch DDL statements.

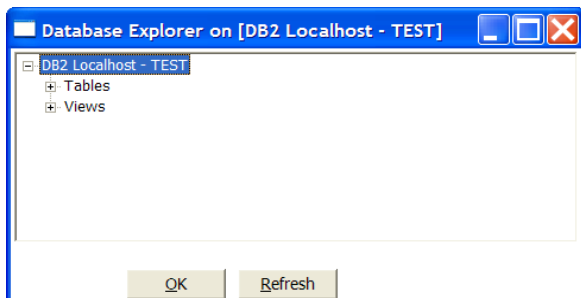
### 4.2. Screen shot



## 5. DATABASE EXPLORER

---

### 5.1. Screen shot



### 5.2. Description

The database explorer allows you to explore the database to which the database connection points. At the moment, it only shows available tables and the catalog and/or schema to which the table belongs.

It is possible to right click on a shown table or view (lowest level in the tree) and select one of the following options:

- ✓ Display the first 100 rows of the table (also available through double-click on table name)
- ✓ Display the first ... lines of the table
- ✓ Show the size (in rows) of the table
- ✓ Show layout of the table
- ✓ Generate the DDL statement (create table ...) for this table
- ✓ Generate the DDL statement (create table ...) for this table on another database connection
- ✓ Show the SQL statement to read from this table (in SQL Editor)



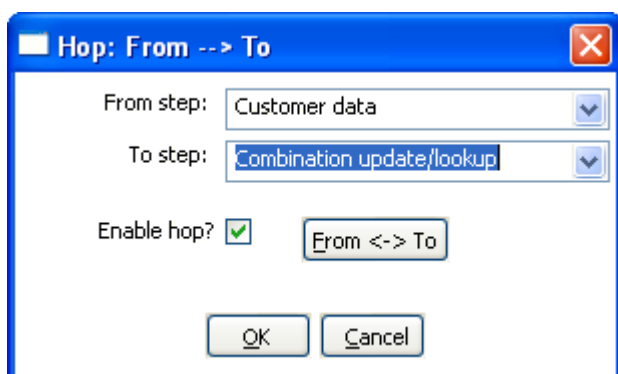
## 6. HOPS

### 6.1. Description

A hop connects one step with another. The direction of the data flow is indicated with an arrow on the graphical view pane. A hop can be enabled or disabled (for testing purposes for example).

When a hop is disabled, the steps downstream of the disabled hop are cut off from any data flowing upstream of the disabled hop. This may lead to unexpected results when editing the downstream steps. For example, if a particular step-type offers a “Get Fields” button, clicking the button may not reveal any of the incoming fields as long as the hop is still disabled.

### 6.2. Screen shot



### 6.3. Creating A Hop

You can easily create a new hop between 2 steps by one of the following options:

- ✓ Dragging on the Graphical View between 2 steps while using the middle mouse button.
- ✓ Dragging on the Graphical View between 2 steps while pressing the SHIFT key and using the left mouse button.
- ✓ Selecting 2 steps in the tree, clicking right and selecting "new hop"
- ✓ Selecting 2 steps in the graphical view (CTRL + left mouse click), right clicking on a step and selecting "new hop"

### 6.4. Splitting A Hop

You can easily insert a new step into a new hop between 2 steps by dragging the step (in the Graphical View) over a hop until the hop becomes drawn in bold. Release the left button and you will be asked if you want to split the hop. This works only with steps that have not yet been connected to another step.

### 6.5. Loops

Loops are not allowed in transformations because Spoon depends heavily on the previous steps to determine the field values that are passed from one step to another. If we would allow loops in transformations we often would get endless loops and undetermined results.

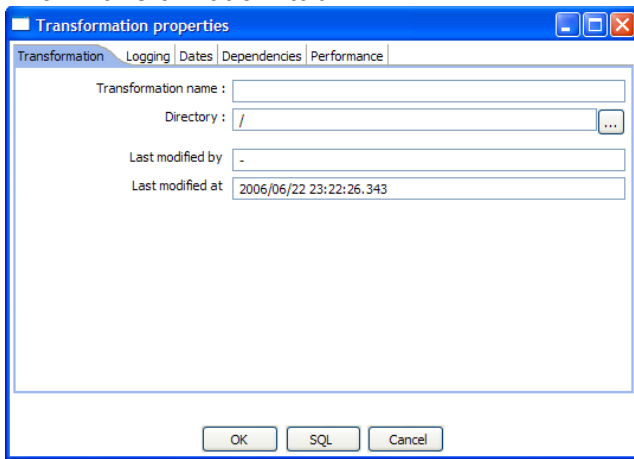
## 7. TRANSFORMATION SETTINGS

### 7.1. Description

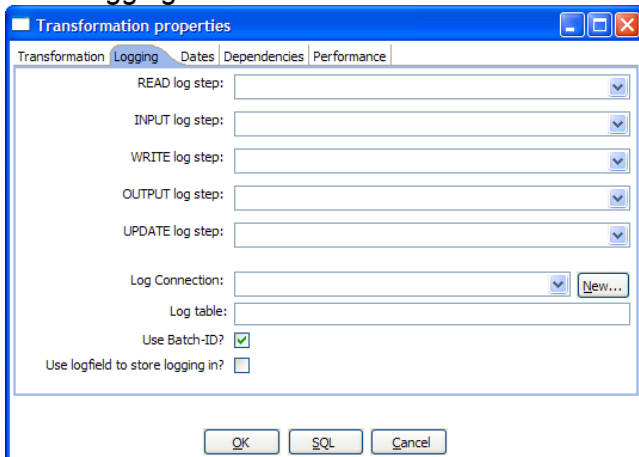
There are a few options that control how a transformation is behaving and how it is logging what it is doing.

### 7.2. Screen shots

*The “Transformation” tab*



*The “Logging” tab*



### The “Dates” tab

**Transformation properties**

Transformation | **Logging** | **Dates** | Dependencies | Performance

Maxdate Connection: None

Maxdate table:

Maxdate field:

Maxdate offset (seconds):

Max. date difference (seconds):

OK SQL Cancel

*The “Dependencies” tab.*

[illegible]

*The “Performance” tab.*

**Transformation properties**


Transformation | Logging | Dates | Dependencies | **Performance**

Nr of rows in rowset :

OK SQL Cancel

## 7.3. Options

<i>Option</i>	<i>Description</i>
Transformation name	the name of the transformation. Required information if you want to save to a repository.
Directory	the directory in the repository in which you want to save the transformation.
READ log step	use the number of read lines from this step to write to the log table. Read means: read from source steps.
INPUT log step	use the number of input lines from this step to write to the log table. Input means: input from file or database.
WRITE log step	use the number of written lines from this step to write to the log table. Written means: written to target steps.
OUTPUT log step	use the number of output lines from this step to write to the log table. Output means: output to file or database.
UPDATE log step	use the number of updated lines from this step to write to the log table. Update means: updated in a database.
Log connection	use this connection to write to a log table.
Log table	specifies the name of the log table (for example L_ETL)
Use Batch-ID?	enable this if you want to have a batch ID in the L_ETL file. Disable for backward compatibility with Spoon/Pan version < 2.0.
Use logfield to store logging in	this option stores the logging text in a CLOB field in the logging table. This allows you to have the logging text together with the run results in the same table. Disable for backward compatibility with Spoon/Pan version < 2.1
Maxdate connection	get the upper limit for a date range on this connection.
Maxdate table	get the upper limit for a date range in this table.
Maxdate field	get the upper limit for a date range in this field.
Maxdate offset	increases the upper date limit with this amount. Use this for example, If you find that the field DATE_LAST_UPD has a maximum value of 2004-05-29 23:00:00, but you know that the values for the last minute are not complete. In this case, simply set the offset to -60.
Maximum date difference	sets the maximum date difference in the obtained date range. This will allow you to limit job sizes.
Dependencies	this table allows you to enter all the dependencies. For example if a dimension is depending on 3 lookup tables, we have to make sure that these lookup tables have not changed. If the values in these lookup tables have changed, we need to extend the date range to force a full refresh of the dimension. The dependencies allow you to look up whether a table has changed in case you have a "data last changed" column in the table.
Number of rows in rowsets	this option allows you to change the size of the buffers between the connected steps in a transformation. You will rarely/never need to change this parameter. Only when you run low on memory it might be an option to lower this parameter.

User manual Last updated: 12-07-2006	Pentaho Data Integration	
	Spoon 2.3.0	

## 7.4. Extra

- ✓ Get dependencies button: Tries to automatically detect dependencies.
- ✓ SQL button: generates the SQL needed to create the logging table and allows you to execute this SQL statement.

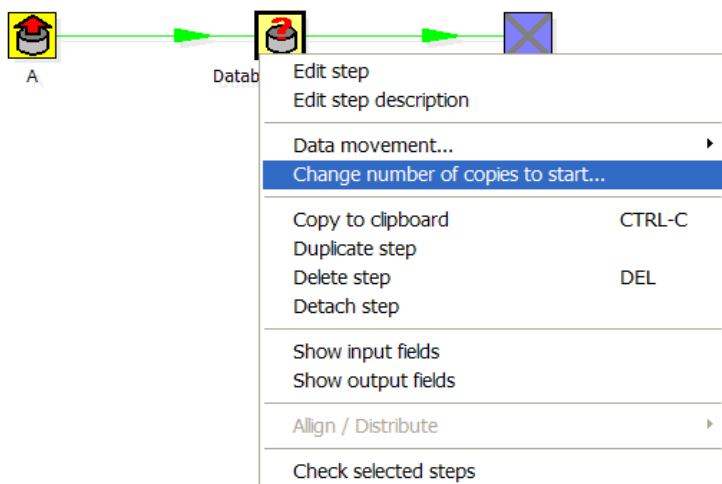
## 8. STEPS

### 8.1. Description

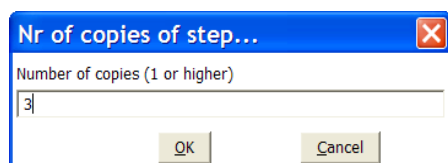
A step is one part of a transformation. Steps can provide you with a wide range of functionality ranging from reading text-files to implementing slowly changing dimensions. Please see below ([7.4. Step Types](#)) for a complete list of all available step-types.

### 8.2. Launching several copies of a step

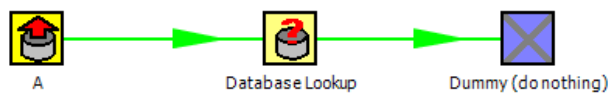
Sometimes it can be useful to launch the same step several times. For example, for performance reasons it can be useful to launch a database lookup step 3 times or more. That is because database connections usually have a certain latency. Launching the same step several times keeps the database busy on different connections, effectively lowering the latency. You can launch several copies of step in a transformation simply by clicking right on a step in the graphical view and then by selecting “change number of copies to start...”



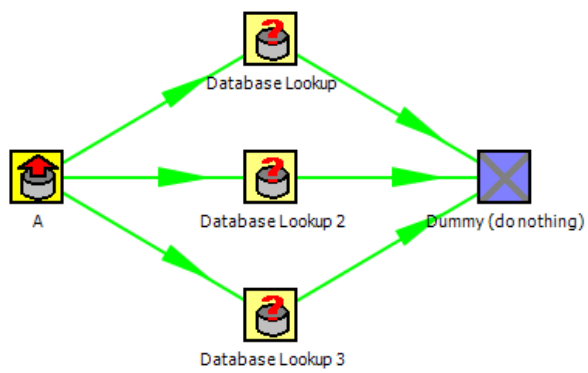
You will get this dialog:



If you enter 3 this will be shown:



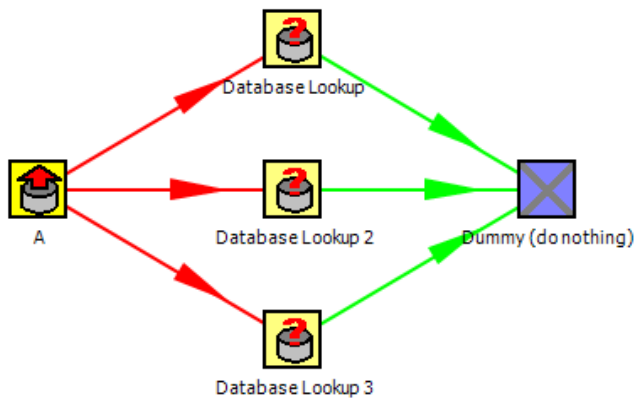
It is technically the equivalent of this:



### 8.3. Distribute or copy?

In the example above green lines are shown between the steps. This indicates that rows are distributed among the target steps. In this case it means that the first row coming from step “A” goes to step “database lookup 1”, the second to “database lookup 2”, the third to “Database lookup 3”, the fourth back to “database lookup 1”, etc.

However, if we right click on step “A”, and select “Copy data”, you will get the hops drawn in red:



“Copy data” means that all rows from step “A” are copied to all 3 the target steps. In this case it means that step “B” gets 3 copies of all the rows that “A” has sent out.

**NOTE:** *Because of the fact that all these steps are run as different threads, the order in which the single rows arrive at step “B” is probably not going to be the same as they left step “A”.*



## 8.4. Use of variables

Variables can be used in various places throughout Pentaho Data Integration, including transformation steps and job entries.

The way to use them is either by grabbing them using step [8.4.45. Get Variable](#) or by specifying meta-data strings like:

- `${VARIABLE}` or
- `%%VARIABLE%%`

Both formats can be used and even mixed, the first is a UNIX derivative, the second is derived from Microsoft Windows.

**NOTE:** Environment variables can also be referenced using this format. A convenient way of setting environment variables in Pentaho Data Integration is by setting them in file `kettle.properties` that can be found in found in directory

`$HOME/.kettle` (Unix/Linux/OSX)

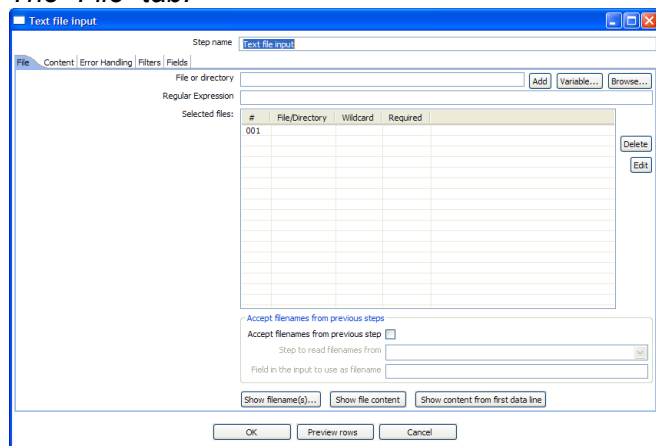
`C:\Documents and Settings\<username>\.kettle\` (Windows)

## 8.5. Step Types

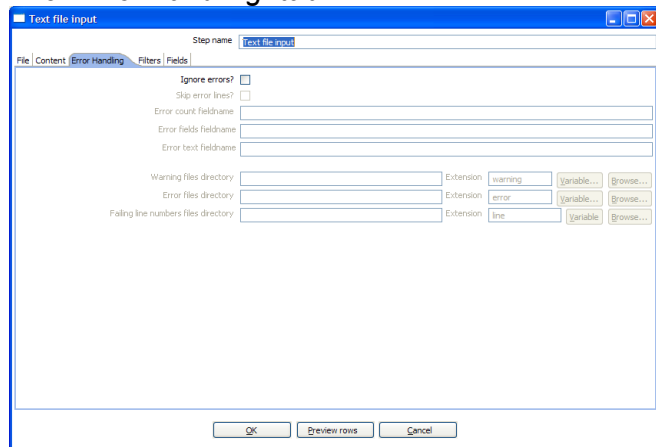
### 8.5.1. Text File Input

#### 8.5.1.1. Screen shots

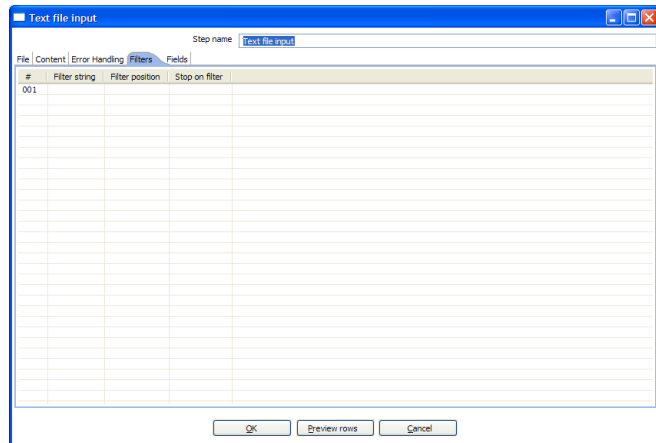
The “File” tab:



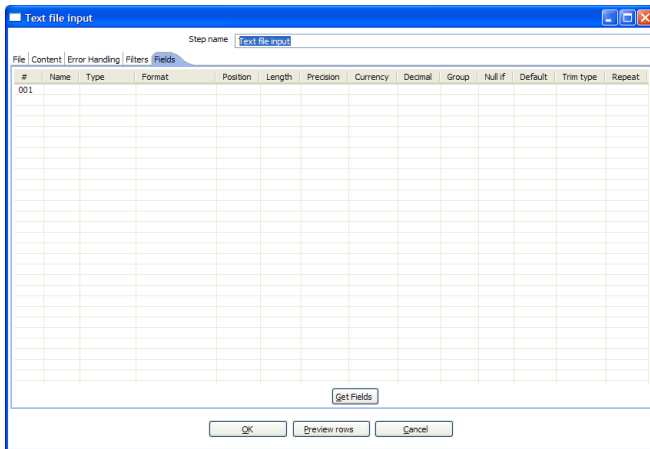
The “Error handling” tab:



The “Filters” tab:



### The “Fields” tab:



#### 8.5.1.2. Icon



#### 8.5.1.3. General description

You can use the *Text File Input* step to read a large number of different text-files.

Most commonly these are Comma Separated Values (CSV files) generated by spreadsheets and the like.

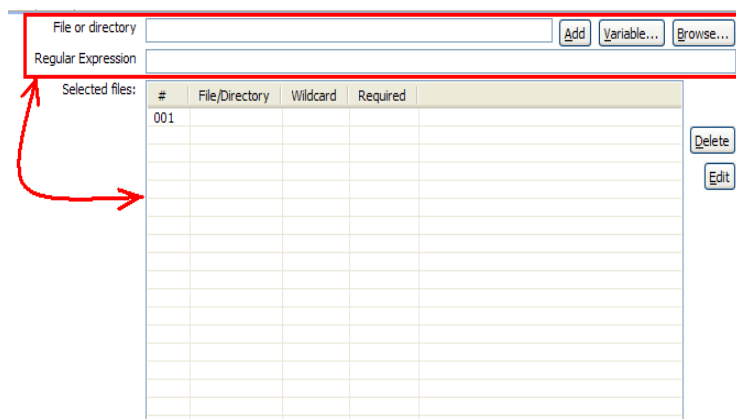
From version 2.1 on you can specify a list of files to read, or a list of directories with wildcards in the form of regular expressions.

Version 2.3. added the ability to accept filenames from a previous step making filename handling more even more generic.

### 8.5.1.4. Options

#### 8.5.1.4.1. Filename specification options

By specifying the filename and pressing the “Add” button, you can add a file to the “selected files” list as seen below:



You can also have this step search for files by specifying a wild card in the form of a regular expression. Regular expressions are more sophisticated than simply using '\*' and '?' wild cards.

Here are a few examples of regular expressions:

Filename	Regular Expression	Files selected
/dirA/	.*userdata.*\..txt	All files in /dirA/ with names containing userdata and ending on .txt
/dirB/	AAA.*	All files in /dirB/ with names starting out with AAA
/dirC/	[A-Z][0-9].*	All files in /dirC/ with names starting with a capital and followed by a digit (A0-Z9)

Option	Description
File or directory	This field specifies the location and/or name of the input text file. <b>NOTE:</b> press the “add” button to add the file/directory/wildcard combination to the list of selected files (grid) below.
Regular expression	Specify the regular expression you want to use to select the files in the directory specified in the previous option.
File type	This can be either CSV or Fixed length. Based on this selection, Spoon will launch a different helper GUI when you press the “get fields” button in the last “fields” tab.
Required	If a file is required and it isn't found, an error is generated. Otherwise, the filename is simply skipped.

#### 8.5.1.4.2. Accept filenames from previous step

Accept filenames from previous steps

Accept filenames from previous step ☒

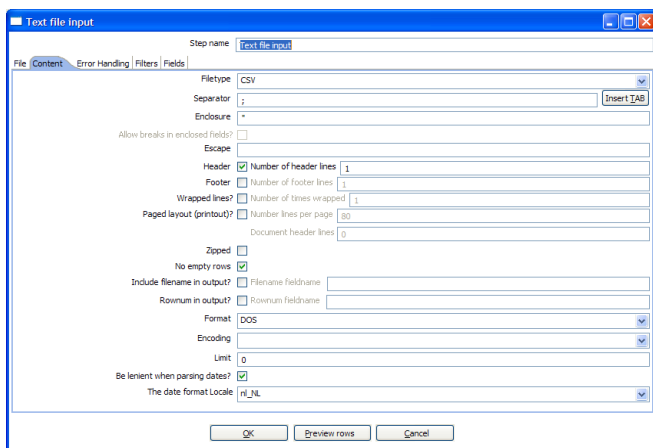
Step to read filenames from

Field in the input to use as filename

This option allows even more flexibility in combination with other steps like “Get Filenames” you can construct your filename and pass it to this step. This way the filename can come from any source: text file, database table, etc.

Option	Description
Accept filenames from previous steps	This enables the option to get filenames from a previous steps.
Step to read filenames from	The step to read the filenames from
Field in the input to use as filename	Text File Input will look in this step to determine the filenames to use.

#### 8.5.1.4.3. Content specification



The content tab allows you to specify the format of the text files that are being read. Here is a list of the options on this tab:

Option	Description
File type	This can be either CSV or Fixed length. Based on this selection, Spoon will launch a different helper GUI when you press the “get fields” button in the last “fields” tab.
Separator	One or more characters that separate the fields in a single line of text. Typically this is ; or a tab.

<i><b>Option</b></i>	<i><b>Description</b></i>
Enclosure	Some fields can be enclosed by a pair of strings to allow separator characters in fields. The enclosure string is optional. If you use repeat an enclosures allow text line 'Not the nine o'clock news.'. With ' the enclosure string, this gets parsed as Not the nine o'clock news.
Escape	Specify an escape character (or characters) if you have escaped characters in your data. If you have \ as an escape character, the text 'Not the nine o'clock news.' (with ' the enclosure) will get parsed as Not the nine o'clock news.
Header & number of header lines	Enable this option if your text file has a header row. (First lines in the file) You can specify the number of times the header lines appears.
Footer & number of footer lines	enable this option if your text file has a footer row. (Last lines in the file) You can specify the number of times the footer row appears.
Wrapped lines & number of wraps	Use this if you deal with <b>data</b> lines that have wrapped beyond a certain page limit. Note that headers & footers are never considered wrapped.
Paged layout & page size & doc header	You can use these options as a last resort when dealing with texts meant for printing on a line printer. Use the number of document header lines to skip introductory texts and the number of lines per page to position the data lines.
Zipped	enable this option if your text file is placed in a Zip archive. <b>NOTE: At the moment, only the first file in the archive is read.</b>
No empty rows	Don't send empty rows to the next steps.
Include filename in output	Enable this if you want the filename to be part of the output.
Filename field name	The name of the field that contains the filename.
Row number in output	Enable this if you want the row number to be part of the output.
Row number field name	The name of the field that contains the row number.
Format	This can be either DOS, UNIX or mixed. UNIX files have lines that are terminated by line feeds. DOS files have lines separated by carriage returns <i>and</i> line feeds. If you specify mixed, no verification is done.
Encoding	Specify the text file encoding to use. Leave blank to use the default encoding on your system. To use Unicode specify UTF-8 or UTF-16. On first use, Spoon will search your system for available encodings.
Limit	Sets the number of lines that is read from the file. 0 means: read all lines.
Be lenient when parsing dates?	Disable this option if you want strict parsing of data fields. In case lenient parsing is enabled, dates like Jan 32 <sup>nd</sup> will become Feb 1 <sup>st</sup> .
The date format Locale	This locale is used to parse dates that have been written in full like "February 2 <sup>nd</sup> , 2006". Parsing this date on a system running in the French (fr_FR) locale would not work because February would be called Février in that locale.

#### 8.5.1.4.4. Error handling

The error handling tab was added to allow you to specify how this step should react when errors occur.

Ignore errors? ☒  
 Skip error lines? ☐  
 Error count fieldname   
 Error fields fieldname   
 Error text fieldname   
  
 Warning files directory  Extension     
 Error files directory  Extension     
 Failing line numbers files directory  Extension

Option	Description
Ignore errors?	Check this option if you want to ignore errors during parsing
Skip error lines	Enable this option if you want to skip those lines that contain errors. Note that you can generate an extra file that will contain the line numbers on which the errors occurred. If lines with errors are not skipped, the fields that did have parsing errors, will be empty (null)
Error count field name	Add a field to the output stream rows. This field will contain the number of errors on the line.
Error fields field name	Add a field to the output stream rows. This field will contain the field names on which an error occurred.
Error text field name	Add a field to the output stream rows. This field will contain the descriptions of the parsing errors that have occurred.
Warnings file directory	When warnings are generated, they will be put in this directory. The name of that file will be <warning_dir>/filename.<date_time>.<warning extension>
Error files directory	When errors occur, they will be put in this directory. The name of that file will be <errorfile_dir>/filename.<date_time>.<errorfile_extension>
Failing line numbers files directory	When a parsing error occur on a line, the line number will be put in this directory. The name of that file will be <errorline_dir>/filename.<date_time>.<errorline extension>

#### 8.5.1.4.5. Filters

On the “Filters” tab you can specify the lines you want to skip in the text file.

#	Filter string	Filter position	Stop on filter
001			

<i><b>Option</b></i>	<i><b>Description</b></i>
Filter string	The string to look for.
Filter position	The position where the filter string has to be at in the line. 0 is the first position in the line. If you specify a value below 0 here, the filter string is searched for in the entire string.
Stop on filter	Specify Y here if you want to stop processing the current text file when the filter string is encountered.



#### 8.5.1.4.6. Fields

<b>Option</b>	<b>Description</b>
Name	name of the field
Type	Type of the field can be either String, Date or Number
Format	See <a href="#">8.4.1.5. Formats</a> for a complete description of format symbols.
Length	For <i>Number</i> : Total number of significant figures in a number; For <i>String</i> : total length of string; For <i>Date</i> : length of <i>printed</i> output of the string (e.g. 4 only gives back the year).
Precision	For <i>Number</i> : Number of floating point digits; For <i>String</i> , <i>Date</i> , <i>Boolean</i> : unused;
Currency	used to interpret numbers like \$10,000.00 or €5.000,00
Decimal	A decimal point can be a "." (10;000.00) or "," (5.000,00)
Grouping	A grouping can be a dot "," (10;000.00) or "." (5.000,00)
Null if	treat this value as NULL
Default	The default value in case the field in the text file was not specified. (empty)
Trim type	trim this field (left, right, both) before processing
Repeat	Y/N: If the corresponding value in this row is empty: repeat the one from the last time it was not empty

### 8.5.1.5. Formats

#### 8.5.1.5.1. Number formats

The information on Number formats was taken from the Sun Java API documentation, to be found here: <http://java.sun.com/j2se/1.4.2/docs/api/java/text/DecimalFormat.html>

<b>Symbol</b>	<b>Location</b>	<b>Localized</b>	<b>Meaning</b>
0	Number	Yes	Digit
#	Number	Yes	Digit, zero shows as absent
.	Number	Yes	Decimal separator or monetary decimal separator
-	Number	Yes	Minus sign
,	Number	Yes	Grouping separator
E	Number	Yes	Separates mantissa and exponent in scientific notation. Need not be quoted in prefix or suffix.
;	Sub pattern boundary	Yes	Separates positive and negative sub patterns
%	Prefix or suffix	Yes	Multiply by 100 and show as percentage
\u2030	Prefix or suffix	Yes	Multiply by 1000 and show as per mille
¤ (\u00A4)	Prefix or suffix	No	Currency sign, replaced by currency symbol. If doubled, replaced by international currency symbol. If present in a pattern, the monetary decimal separator is used instead of the decimal separator.
'	Prefix or suffix	No	Used to quote special characters in a prefix or suffix, for example, "'#'" formats 123 to "#123". To create a single quote itself, use two in a row: "' o'clock".

#### Scientific Notation

In a pattern, the exponent character immediately followed by one or more digit characters indicates scientific notation. Example: "0.###E0" formats the number 1234 as "1.234E3".

#### 8.5.1.5.2. Date formats

The information on Date formats was taken from the Sun Java API documentation, to be found here: <http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html>

<b>Letter</b>	<b>Date or Time Component</b>	<b>Presentation</b>	<b>Examples</b>
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

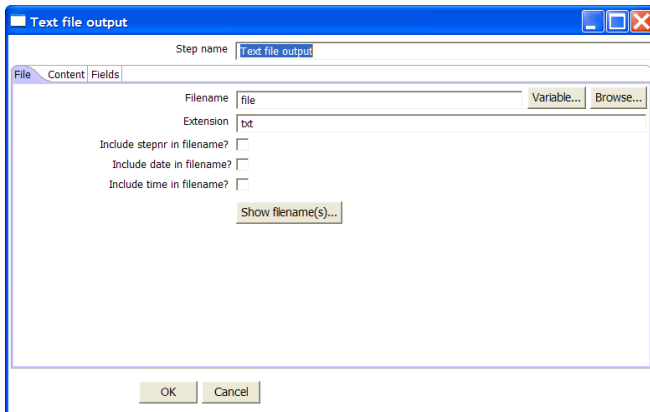
#### 8.5.1.6. Extras

<i><b>Function/Button</b></i>	<i><b>Description</b></i>
Show filenames	This option shows a list of all the files selected. Please note that if the transformation is to be run on a separate server, the result might be incorrect.
Show file content	The “View” button shows the first lines of the text-file. Make sure that the file-format is correct. When in doubt, try both DOS and UNIX formats.
Show content from first data line	This button helps you in positioning the data lines in complex text files with multiple header lines, etc.
Get fields	This button allows you to guess the layout of the file. In case of a CSV file, this is done pretty much automatically. When you selected a file with fixed length fields, you need to specify the field boundaries using a wizard.
Preview rows	Press this button to preview the rows generated by this step.

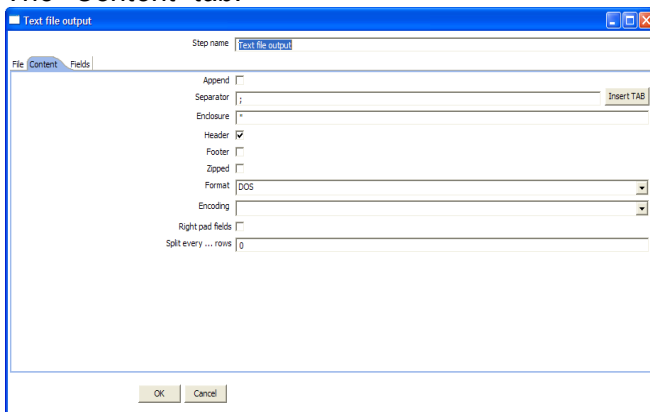
## 8.5.2. Text File Output

### 8.5.2.1. Screenshots

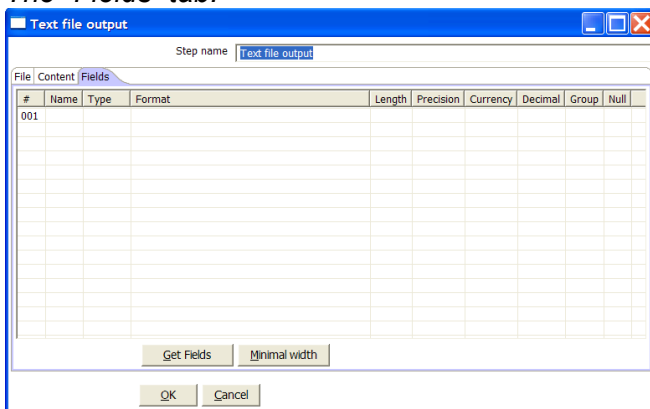
The “File” tab:



The “Content” tab:



The “Fields” tab:



#### 8.5.2.2. Icon



#### 8.5.2.3. General Description

You can use the *Text File Output* step to export data to text file format.

Most commonly these are Comma Separated Values (CSV files) that can be read by spreadsheets and the like.

#### 8.5.2.4. Options

<b>Option</b>	<b>Description</b>
Step name	Name of the step. This name has to be unique in a single transformation.
Filename	This field specifies the filename and location of the output text file.
Extension	Adds a point and the extension to the end of the filename. (.txt)
Include stepnr in filename	If you run the step in multiple copies (see <a href="#">7.2. Launching several copies of a step</a> ), the copy number is included in the filename, before the extension. (_0).
Include date in filename	Includes the system date in the filename. (_20041231).
Include time in filename	Includes the system date in the filename. (_235959).
Append	Check this if you want to append lines to the end of the specified file.
Separator	Here you can specify the character that separates the fields in a single line of text. Typically this is ; or a tab.
Enclosure	A pair of strings can enclose some fields. This allows separator characters in fields. The enclosure string is optional.
Header	Enable this option if you want the text file to have a header row. (First line in the file).
Footer	Enable this option if you want the text file to have a footer row. (Last line in the file).
Zipped	Check this box if you want the text file(s) to be zipped. <b>NOTE: At the moment, only one file is placed in a single archive.</b>
Format	This can be either DOS or UNIX. UNIX files have lines separated by linefeeds. DOS files have lines separated by carriage returns <i>and</i> line feeds.
Encoding	Specify the text file encoding to use. Leave blank to use the default encoding on your system. To use Unicode specify UTF-8 or UTF-16. On first use, Spoon will search your system for available encodings.
Right pad fields	Add spaces to the end of the fields (or remove characters at the end) until they have the specified length.
Split every ... rows	If this number N is larger than zero, split the resulting text-file into multiple parts of N rows.

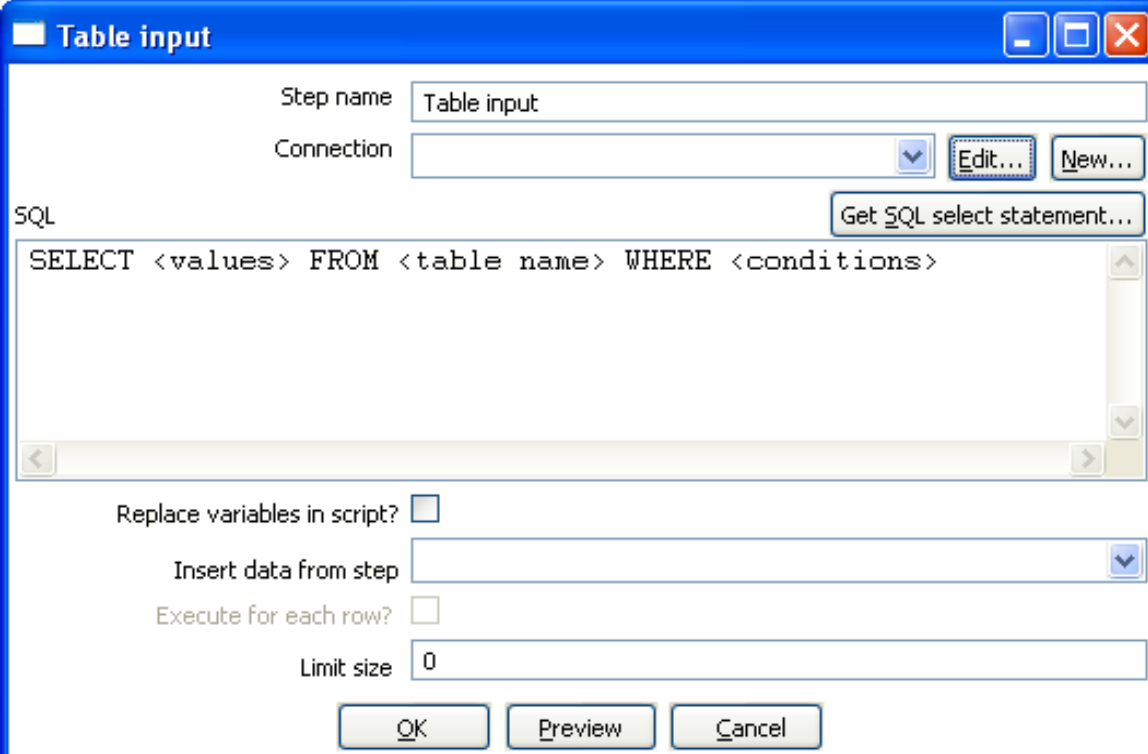
Option	Description
Fields	<ul style="list-style-type: none"> <li>✓ Name: name of the field</li> <li>✓ Type: Type of the field can be either String, Date or Number</li> <li>✓ Format: See <a href="#">7.4.1.5. Formats</a> for a complete description of format specifiers</li> <li>✓ Length:               <ul style="list-style-type: none"> <li>○ Number: Total number of significant figures in a number</li> <li>○ String: total length of string;</li> <li>○ Date: length of <i>printed</i> output of the string (e.g. 4 only gives back the year).</li> </ul> </li> <li>✓ Precision:               <ul style="list-style-type: none"> <li>○ Number: Number of floating point digits;</li> <li>○ String: unused;</li> <li>○ Date: unused.</li> </ul> </li> <li>✓ Currency: Symbol used to represent currencies like \$10,000.00 or €5.000,00</li> <li>✓ Decimal: A decimal point can be a "." (10,000.00) or "," (5.000,00)</li> <li>✓ Grouping: A grouping can be a "," (10,000.00) or "." (5.000,00)</li> <li>✓ Null: If the value of the field is null, insert this string into the text-file</li> </ul>

#### 8.5.2.5. Extras

Function/Button	Description
Show filenames	This option shows a list of the files the will be generated. <b>NOTE:</b> <i>This is a simulation and sometimes depends on the number of rows in each file, etc.</i>
Minimal width	Alter the options in the fields tab in such a way that the resulting width of lines in the text file is minimal. So instead of save 0000001, we write 1, etc. String fields will no longer be padded to their specified length.

## 8.5.3. Table input

### 8.5.3.1. Screenshot



### 8.5.3.2. Icon



### 8.5.3.3. General description

This step is used to read information from a database, using a connection and SQL. Basic SQL statements are generated automatically.

#### 8.5.3.4. Options

Option	Description
Step name	Name of the step. This name has to be unique in a single transformation.
Connection	The database connection used to read data from.
SQL	The SQL statement used to read information from the database connection.
Insert data from step	Specify the input step name where we can expect information to come from. This information can then be inserted into the SQL statement. The locators where we insert information is indicated by ? (question marks).
Limit	Sets the number of lines that is read from the database. 0 means: read all lines.

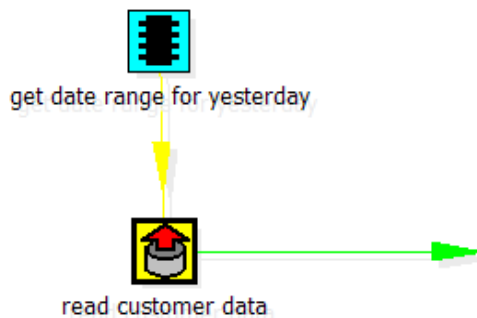
#### 8.5.3.5. Example

Consider for example the following SQL statement:

```
SELECT * FROM customers WHERE changed_date BETWEEN ? AND ?
```

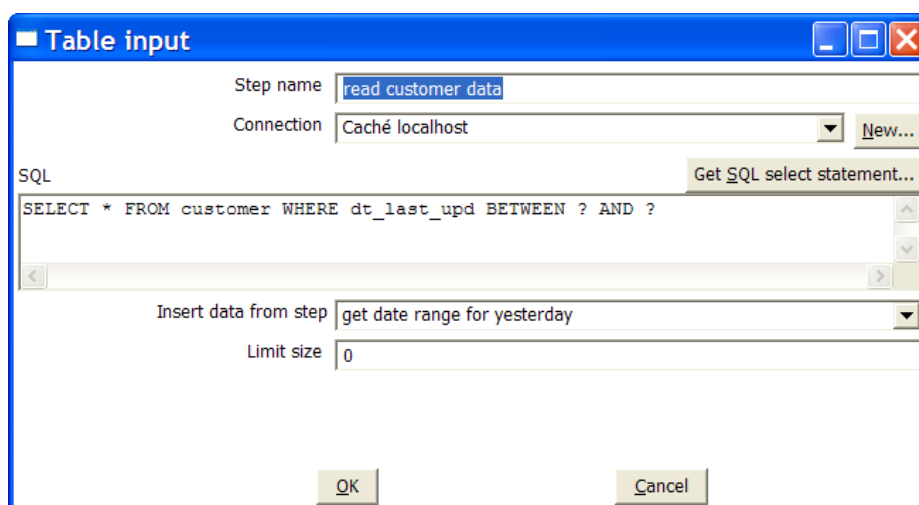
This statement needs 2 dates that are read on the "Insert data from" step.

**NOTE:** The dates can be provided using the "Get System Info" step type. For example if you want to read all customers that have had their data changed yesterday, you might do it like this:



The "read customer data" step looks like this:





**Table input**

Step name: read customer data

Connection: Cache localhost

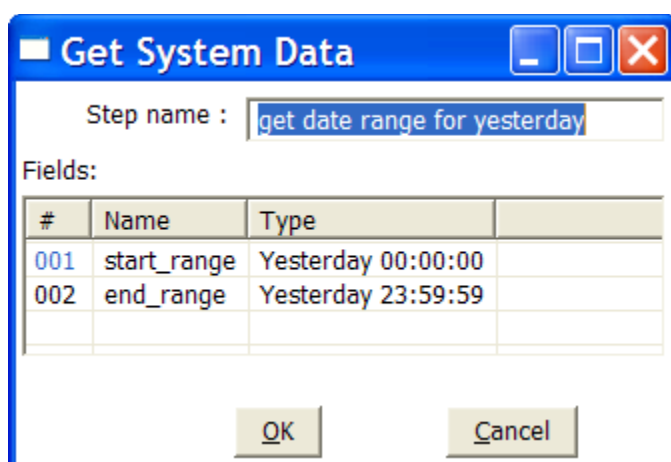
SQL: SELECT \* FROM customer WHERE dt\_last\_upd BETWEEN ? AND ?

Insert data from step: get date range for yesterday

Limit size: 0

OK Cancel

And the “get date range for yesterday” looks like this:



**Get System Data**

Step name : get date range for yesterday

Fields:

#	Name	Type
001	start_range	Yesterday 00:00:00
002	end_range	Yesterday 23:59:59

OK Cancel

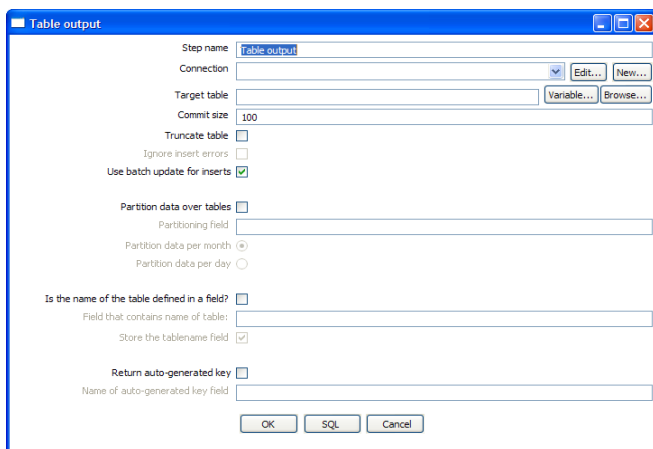
See also: Get System Data step [7.4.16. Get System Info](#).

#### 8.5.3.6. Extras

Function/Button	Description
Preview	This option previews this step. It is done by preview of a new transformation with 2 steps: this one and a Dummy step. You can see the detailed logging of that execution by clicking on the logs button in the the preview window.

## 8.5.4. Table output

### 8.5.4.1. Screen dump



### 8.5.4.2. Icon



### 8.5.4.3. General description

This step type allows you to store information in a database table.

### 8.5.4.4. Options

Option	Description
Step name	Name of the step. This name has to be unique in a single transformation.
Connection	The database connection used to write data to.
Target table	The name of the table to write data to.
Commit size	Use transactions to insert rows in the database table. Commit the connection every N rows if N is larger than 0. Otherwise, don't use transactions. (Slower) <b>NOTE: Transactions are not supported on all database platforms.</b>
Truncate table	Select this if you want the table to be truncated <i>before</i> the first row is inserted into the table.
Ignore insert errors	Makes Kettle ignore all insert errors such as violated primary keys. A maximum of 20 warnings will be logged however. This option is not available for batch inserts.
Use batch update for inserts	Enable this option if you want to use batch inserts. This feature groups inserts statements to limit round trips to the database. This is the fastest option and is enabled by default.

<i>Option</i>	<i>Description</i>
Partition data over tables	<p>Use this options to split the data over multiple tables. For example instead of inserting all data into table SALES, put the data into tables SALES_200510, SALES_200511, SALES_200512, ...</p> <p>Use this on systems that don't have partitioned tables and/or don't allow inserts into UNION ALL views or the master of inherited tables.</p> <p>The view SALES allows you to report on the complete sales:</p> <pre> CREATE OR REPLACE VIEW SALES AS SELECT * FROM SALES_200501 UNION ALL SELECT * FROM SALES_200502 UNION ALL SELECT * FROM SALES_200503 UNION ALL SELECT * FROM SALES_200504 ... </pre>
Is the name of the table defined in a field.	<p>Use these options to split the data over one or more tables. The name of the target table is defined in the field you specify. For example if you store customer data in the field gender, the data might end up in tables M and F (Male and Female).</p> <p>There is an option to exclude the field containing the tablename from being inserted into the tables.</p>
Return auto-generated key	Check this if you want to get back the key that was generated by inserting a row into the table.
Name of auto-generated key field	Specify the name of the new field in the output rows that will contain the auto-generated key.

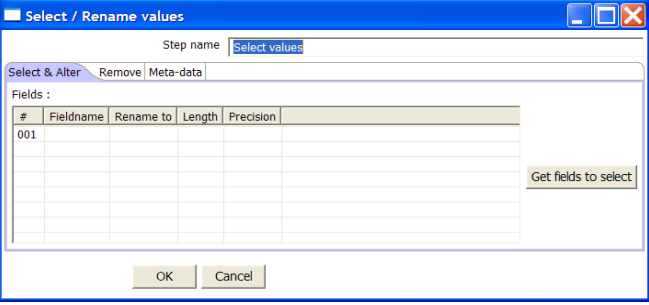
#### 8.5.4.5. Extras

<i>Function/Button</i>	<i>Description</i>
SQL	Generate the SQL to create the output table automatically.
Check	Verifies that all fields are available in the target table and vice-versa.

## 8.5.5. Select values

### 8.5.5.1. Screen dump

The “Select & Alter” tab:



Step name: Select values

Select & Alter | Remove | Meta-data

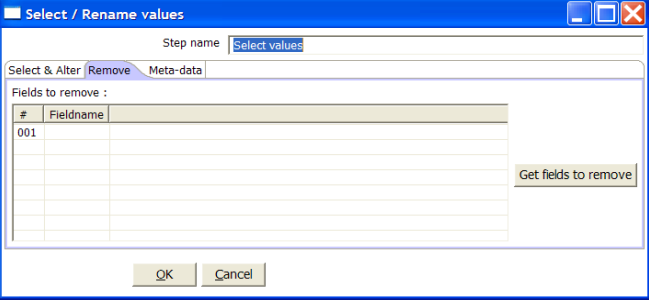
Fields :

#	Fieldname	Rename to	Length	Precision
001				

Get fields to select

OK Cancel

The “Remove” tab:



Step name: Select values

Select & Alter | Remove | Meta-data

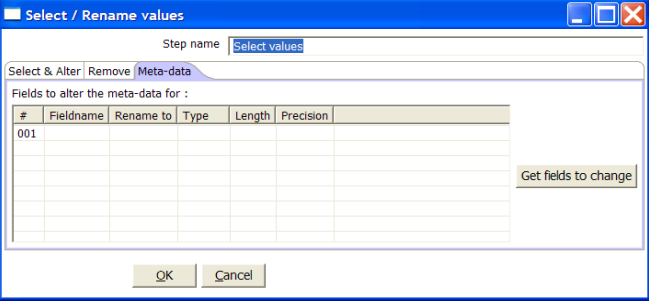
Fields to remove :

#	Fieldname
001	

Get fields to remove

OK Cancel

The “Meta-data” tab:



Step name: Select values

Select & Alter | Remove | Meta-data

Fields to alter the meta-data for :

#	Fieldname	Rename to	Type	Length	Precision
001					

Get fields to change

OK Cancel

### 8.5.5.2. Icon



#### 8.5.5.3. General description

This step type is useful to

- ✓ Select fields
- ✓ Rename fields
- ✓ Specify the length and/or precision of the fields

These are the functions of the 3 different tabs:

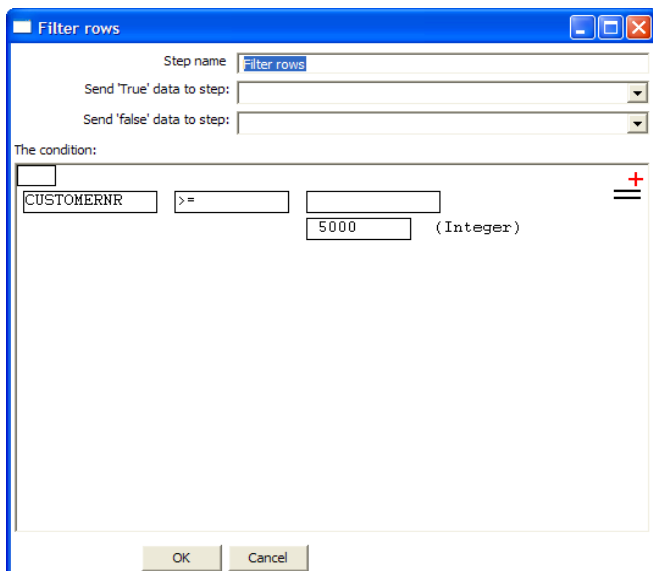
- Select & Alter: Specify the exact order and name in which the fields have to be placed in the output rows.
- Remove: Specify the fields that have to be removed from the output rows.
- Meta-data: Change the name, type, length and precision (the meta-data) of one or more fields.

#### 8.5.5.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Fields of the table:**
  - Fieldname: The fieldname to select or change
  - Rename to: leave blank if you don't want to rename
  - Length: enter number to specify the length. (-1: no length specified)
  - Precision: enter number to specify the precision. (-1: no precision specified)

## 8.5.6. Filter rows

### 8.5.6.1. Screenshot



### 8.5.6.2. Icon



### 8.5.6.3. General description

This step type allows you to filter rows based upon conditions and comparisons.

Once this step is connected to a previous step (one or more and receiving input), you can simply click on the “<field>”, “=” and “<value>” areas to construct a condition.

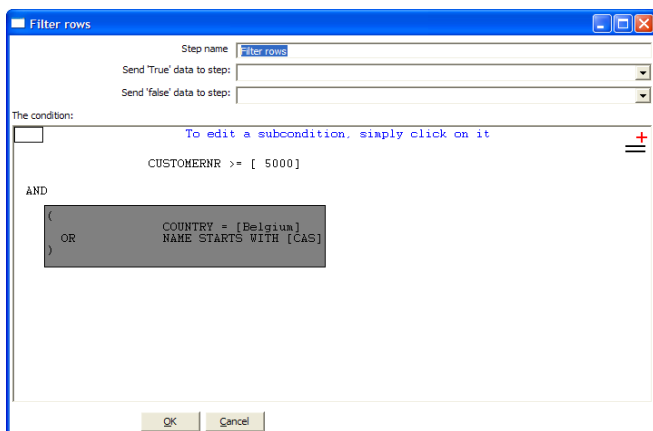
In the example in the screenshot, it is possible to click on the following icon to add more conditions:



It will convert the original condition to a subcondition and add one more.

A subcondition can be edited simply by clicking on it. (going down one level into the condition tree)

For example, this is a more complex example:

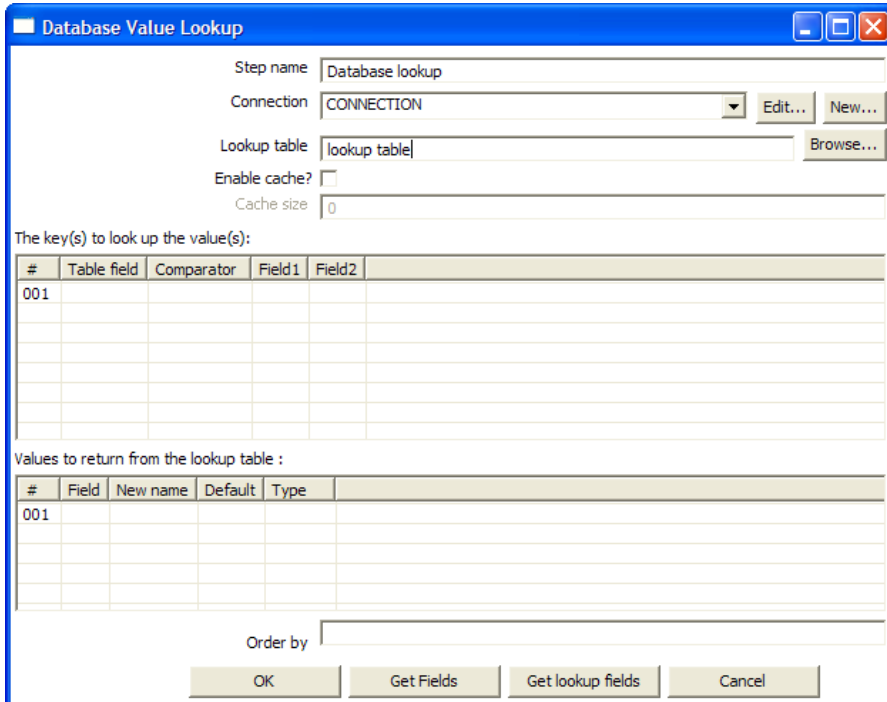


#### 8.5.6.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Send “true” data to step:** The rows for which the condition specified evaluates to true are send to this step.
- ✓ **Send “false” data to step:** The rows for which the condition specified evaluates to false are send to this step.

## 8.5.7. Database lookup

### 8.5.7.1. Screen dump



### 8.5.7.2. Icon



### 8.5.7.3. General description

This step type allows you to look up values in a database table.

### 8.5.7.4. Options

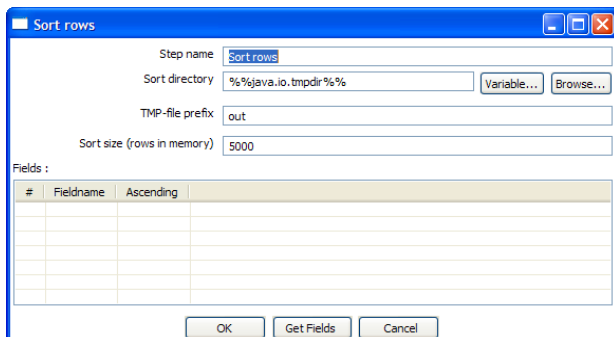
- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Connection:** The database connection used to write data to.
- ✓ **Lookup table:** The name of the table where we do the lookup.
- ✓ **Enable cache:** This option caches database lookups. This means that we expect the database to return the same value all the time for a certain lookup value.

**IMPORTANT:** *if other processes are changing values in the table where you do the lookup, it might be unwise to cache values. However, in all other cases, enabling this option can seriously increase the performance because database lookups are relatively slow. If you find that you can't use the cache, consider launching several copies of this step at the same time. This will keep the database busy via different connections. To see how to do this, please see [7.2. Launching several copies of a step](#)*



## 8.5.8. Sort rows

### 8.5.8.1. Screenshot



### 8.5.8.2. Icon



### 8.5.8.3. General description

This step type sorts rows based upon the fields you specify and whether or not they should be sorted ascending or descending.

**NOTE:** Kettle has to sort rows using temporary files when the number of rows exceeds 5000.

### 8.5.8.4. Options

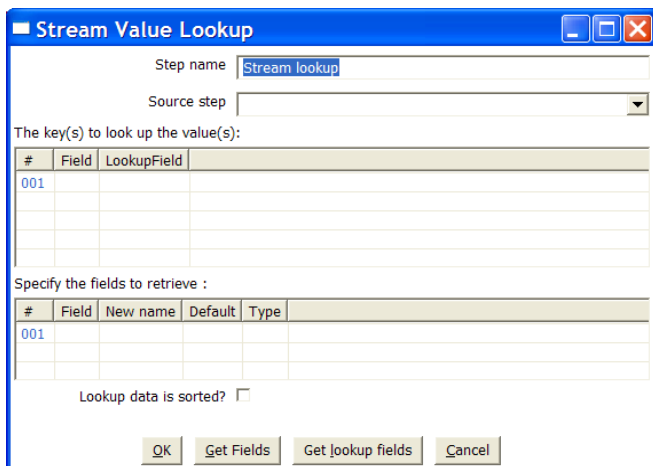
- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Sort directory:** This is the directory in which the temporary files are stored in case it is needed. The default is the standard temporary directory for the system.
- ✓ **Sort size:** The more rows you can store in memory, the faster the sort gets. This is because less temporary files need to be used and less I/O is generated.
- ✓ **The TMP-file prefix:** Choose a recognizable prefix in order to recognize the files when they show up in the temp directory.
- ✓ A list of fields and whether they should be sorted ascending or not.

### 8.5.8.5. Extras

- ✓ The “Get fields” button inserts the fields into the “Fields” grid if the step is connected to the previous ones.

## 8.5.9. Stream lookup

### 8.5.9.1. Screenshot



The screenshot shows the 'Stream Value Lookup' dialog box. It has a title bar with standard window controls. Inside, there's a 'Step name' field containing 'Stream lookup' and a 'Source step' dropdown menu. Below these is a section titled 'The key(s) to look up the value(s):' followed by a table with columns '#', 'Field', and 'LookupField'. The first row is pre-filled with '001'. Below this is another section titled 'Specify the fields to retrieve :' followed by a table with columns '#', 'Field', 'New name', 'Default', and 'Type'. The first row is pre-filled with '001'. At the bottom, there's a checkbox labeled 'Lookup data is sorted?' which is unchecked. At the very bottom are four buttons: 'OK', 'Get Fields', 'Get lookup fields', and 'Cancel'.

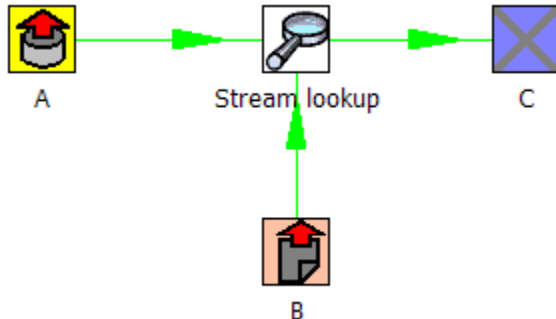
### 8.5.9.2. Icon



### 8.5.9.3. General description

This step type allows you to look up data using information coming from other steps in the transformation. The data coming from the “Source step” is first read into memory and is then used to look up data from the main stream.

For example, this transformation adds information coming from a text-file to data coming from a database table:



The fact that we use information from B to do the lookups is indicated by the option: “Source step” (see below):

Source step

#### 8.5.9.4. Options

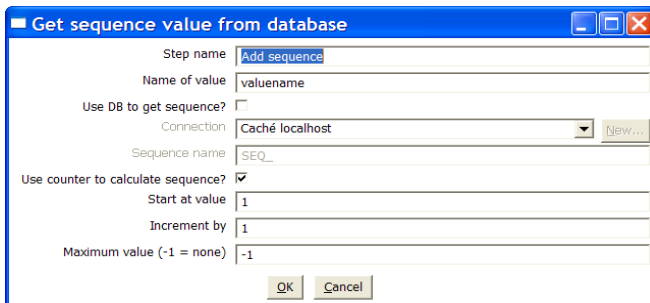
- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Source step:** This is the step name where the lookup data is coming from
- ✓ The key(s) to lookup value(s): Allows you to specify the names of the fields that are used to lookup values. Values are always searched using the “equal” comparison.
- ✓ **Fields to retrieve:** You can specify the names of the fields to retrieve here, as well as the default value in case the value was not found or a new fieldname in case you didn’t like the old one.

#### 8.5.9.5. Extra

- ✓ **“Get Fields”:** This will automatically fill in the names of all the available fields on the source side (A). You can then delete all the fields you don’t want to use for lookup.
- ✓ **“Get lookup fields”:** This will automatically insert the names of all the available fields on the lookup side (B). You can then delete the fields you don’t want to retrieve.

## 8.5.10. Add sequence

### 8.5.10.1. Screen dump



### 8.5.10.2. Icon



### 8.5.10.3. General description

Steps based on this step type will add a sequence to the stream. A sequence is an ever-changing integer value with a certain start and increment value.

You can either use a database (Oracle) sequence to determine the value of the sequence, or let Kettle decide.

**NOTE:** Kettle sequences are only unique when used in the same transformation. Also, they are not stored, so the values start back at the same value every time the transformation is launched.

### 8.5.10.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Name of value:** Name of the new sequence value that is added to the stream.
- ✓ **Use DB to get sequence:** Enable this option if you want the sequence to be driven by a database sequence.
  - **Connection name:** choose the name of the connection on which the database sequence resides.
  - **Sequence name:** allows you to enter the name of the database sequence.
- ✓ **Use counter to calculate sequence:** Enable this option if you want the sequence to be generated by Kettle.
  - **Start at:** give the start value of the sequence.
  - **Increment by :** give the increment of the sequence.
  - **Maximum value:** this is the maximum value after which the sequence will start back at the start value (Start At).

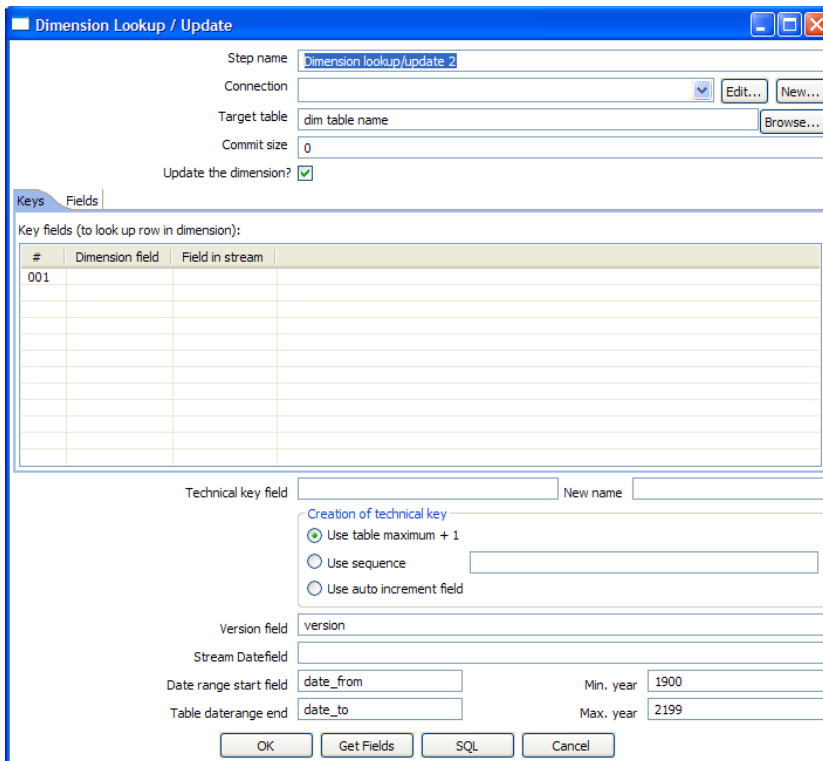
### Examples:

```
Start at = 1, increment by = 1, maximum value = 3
→ This will produce: 1, 2, 3, 1, 2, 3, 1, 2, 3, ...

Start at = 0, increment by = -1, maximum value = -5
→ This will produce: 0, -1, -2, -3, -4, -5, 0, -1, -2, ...
```

## 8.5.11. Dimension lookup/update

### 8.5.11.1. Screenshots



### 8.5.11.2. Icon



### 8.5.11.3. General description

This step type allows you to implement Ralph Kimball's slowly changing dimension for both types: Type I (insert) and Type II (update). Not only can you use this dimension for updating a dimension table, it can also be used for looking up values in a dimension.

In our dimension implementation each entry in the dimension table has the following fields:

1. **Technical key:** This is the primary key of the dimension.
2. **Version field:** Shows the version of the dimension entry (a revision number).
3. **Start of date range:** This is the fieldname containing the validity starting date.
4. **End of date range:** this is the fieldname containing the validity ending date.
5. **Keys:** These are the keys used in your source systems. For example: customer numbers, product id, etc.
6. **Fields:** These fields contain the actual information of a dimension.

As a result of the lookup or update operation of this step type, a field is added to the stream containing the technical key of the dimension. In case the field is not found, the value of the dimension entry for not found (0 or 1, based on the type of database) is returned.

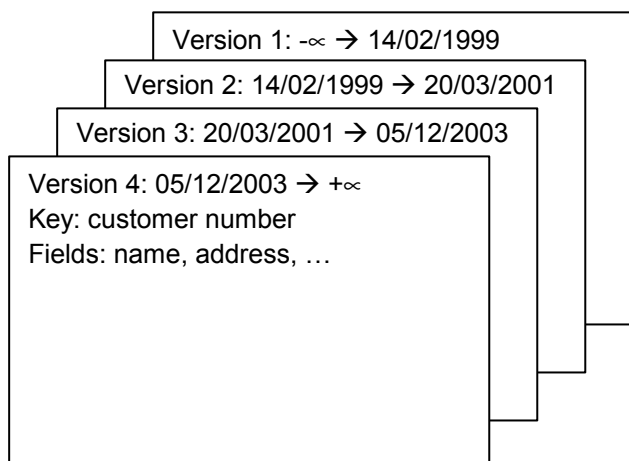
**NOTE:** This dimension entry is added automatically to the dimension table when the update is first run.

#### 8.5.11.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Connection:** Name of the database connection on which the dimension table resides.
- ✓ **Target table:** Name of the dimension table.
- ✓ **Commit size:** Setting this to 10 will generate a commit every 10 inserts or updates.
- ✓ **Update dimension:** Check this option if you want to update the dimension based on the information in the input stream. If this option is not enabled, the dimension only does lookups and only adds the technical key field to the streams.
- ✓ **Technical Key field:** This indicates the primary key of the dimension. It is also referred to as Surrogate Key.
- ✓ **Technical key, new name:** Rename the technical key after a lookup, for example, if you need to lookup different types of products like ORIGINAL\_PRODUCT\_TK, REPLACEMENT\_PRODUCT\_TK, ...
- ✓ **Creation of technical key:** How is the technical key generated, options which are not available for your connection will be grayed out:
  - ✓ Use table maximum + 1: A new technical key will be created from the maximum key in the table. Note that the new maximum is always cached, so that the maximum does not need to be calculated for each new row.
  - ✓ Use sequence: Specify the sequence name if you want to use a database sequence on the table connection to generate the technical key (typical for Oracle e.g.).
  - ✓ Use auto increment field: Use an auto increment field in the database table to generate the technical key (typical for DB2 e.g.).
- ✓ **Version field:** specifies the name of the field to store the version (revision number) in.
- ✓ **Stream date field:** If you have the date at which the dimension entry was last changed, you can specify the name of that field here. It allows the dimension entry to be accurately described for what the date range concerns. If you don't have such a date, the system date will be taken.
- ✓ **Date range start and end fields:** Specify the names of the dimension entries validity range.
- ✓ **Keys:** Specify the names of the keys in the stream and in the dimension table. This will enable the step to do the lookup.
- ✓ **Fields:** For each of the fields you need to have in the dimension, you can specify whether you want the values to be updated (for all versions, this is a Type II operation) or you want to have the values inserted into the dimension as a new version. In the example we used in the screenshot the birth date is something that's not variable in time, so if the birth date changes, it means that it was wrong in previous versions. It's only logical then, that the previous values are corrected in all versions of the dimension entry.

#### 8.5.11.5. Extra

- ✓ **Get fields:** fills in all the available fields on the input stream, except for the keys you specified.
- ✓ **SQL:** generates the SQL to build the dimension and allows you to execute this SQL.



Each dimension entry can be represented by a stack of papers containing the information valid during a certain period of time.

*Insert* then means that we add a new piece of paper containing the new information.

(Type II)

*Punch through* means that we overwrite certain data on all pieces of paper for that certain customer number. (Type I)

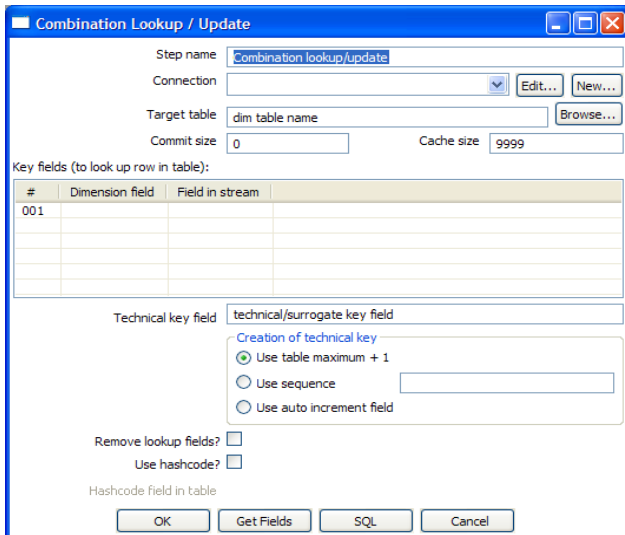
#### 8.5.11.6. Remarks

- ✓ **For the Stream date field:** Consider adding an extra date field from System Info if you don't want the date ranges to be different all the time. For example if you have extracts from a source system being done every night at midnight, consider adding date "Yesterday 23:59:59" as a field to the stream by using a Join step.
- ✓ **For the "Date range start and end fields":** You can only a year in these fields, not a timestamp. If you enter a year YYYY (e.g. 2100), it will be used as timestamp "YYYY-01-01 00:00:00.00" in the dimension table.



## 8.5.12. Combination lookup/update

### 8.5.12.1. Screenshot



### 8.5.12.2. Icon



### 8.5.12.3. General description

This step allows you to store information in a junk-dimension table, and can possibly also be used to maintain Kimball pure Type 1 dimensions.

In short what it will do is:

1. Lookup combination of business key field1... field $n$  from the input stream in a dimension table;
2. If this combination of business key fields exists, return its technical key (surrogate id);
3. If this combination of business key doesn't exist yet, insert a row with the new key fields and return its (new) technical key;
4. Put all input fields on the output stream including the returned technical key, but remove all business key fields if "remove lookup fields" is true.

So what this step does is create/maintain a technical key out of data with business keys. After passing through this step all of the remaining data changes for the dimension table can be made as updates, as either a row for the business key already existed or was created.

This step will only maintain the key information, you will still need to update the non-key information in the dimension table, e.g. by putting an update step (based on technical key) after the combination update/lookup step.

Kettle will store the information in a table where the primary key is the combination of the business key fields in the table. Because this process can be very slow in case you have a large number of fields, Kettle also supports a "hash code" field representing all fields in the dimension. This can speed up lookup performance dramatically while limiting the fields to index to 1.

#### 8.5.12.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Connection:** Name of the database connection on which the dimension table resides.
- ✓ **Target table:** Name of the dimension table.
- ✓ **Commit size:** Setting this to 10 will generate a commit every 10 inserts or updates.
- ✓ **Cache size:** Size of the 'Least-Recently-Used' cache (0 means cache all rows), caching cannot be disabled. This should be set to 0 if you have the memory for caching available, if you process large files consider setting it to a larger number (e.g. 9999).
- ✓ **Key fields:** Specify all the fields you want to store in the dimension.
- ✓ **Technical Key field:** This indicates the primary key of the dimension. It is also referred to as Surrogate Key.
- ✓ **Creation of technical key:** How is the technical key generated, options which are not available for your connection will be greyed out:
  - ✓ Use table maximum + 1: A new technical key will be created from the maximum key in the table. Note that the new maximum is always cached, so that the maximum does not need to be calculated for each new row.
  - ✓ Use sequence: Specify the sequence name if you want to use a database sequence on the table connection to generate the technical key (typical for Oracle e.g.).
  - ✓ Use auto increment field: Use an auto increment field in the database table to generate the technical key (typical for DB2 e.g.).
- ✓ **Remove lookup fields:** Enable this option if you want to remove all the lookup fields from the input stream in the output. The only extra field added is then the technical key.
- ✓ **Use hash code:** This option allows you to generate a hash code, representing all values in the key fields in a numerical form (a signed 64 bit integer). This hash code has to be stored in the table.

#### 8.5.12.5. Extra

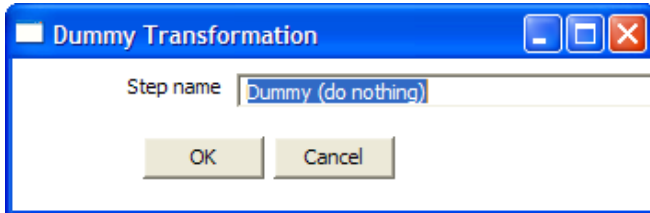
- ✓ **Get fields:** Fills in all the available fields on the input stream, except for the keys you specified.
- ✓ **SQL:** Generates the SQL to build the dimension and allows you to execute this SQL.

#### 8.5.12.6. Remarks

- ✓ The Combination Lookup/Update step assumes that the dimension table it maintains is **not** updated concurrently by other transformations/applications. When you use e.g. the "Table Max + 1" method to create the technical keys the step will not always go to the database to retrieve the next highest technical key. The technical will be cached locally, so if multiple transformations would update the dimension table simultaneously you will most likely get errors on duplicate technical keys. Using a sequence or an auto increment technical key to generate the technical key it is still not advised to concurrently do updates to a dimension table because of possible conflicts between transformations.
- ✓ It is assumed that the technical key is the primary key of the dimension table or at least has a unique index on it. It's not 100% required but if a technical key exists multiple times in the dimension table the result for the Combination Lookup/Update step is unreliable.

### 8.5.13. Dummy (do nothing)

#### 8.5.13.1. Screenshot

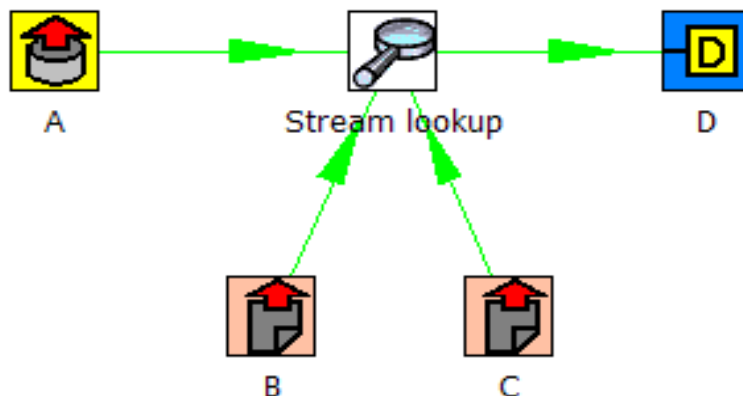


#### 8.5.13.2. Icon

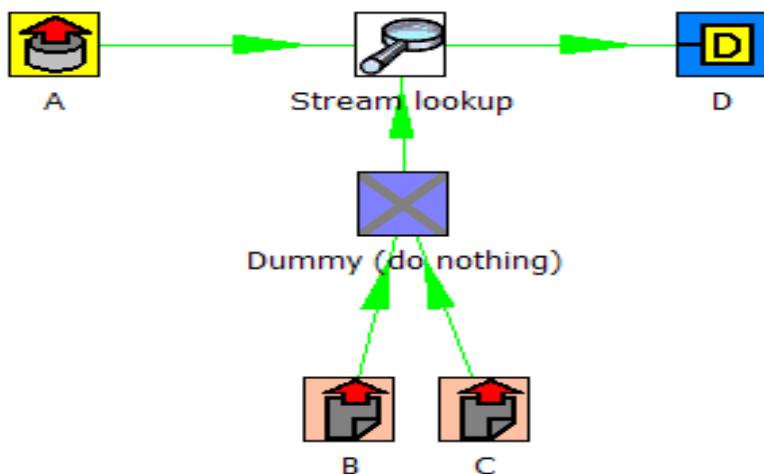


#### 8.5.13.3. General description

This step doesn't do anything. Its main function is perform as a placeholder in case you want to test something. For example, to have a transformation, you need at least 2 steps connected to each other. If you want to test for example a test file input step, you can connect it to a dummy step. The placeholder function also comes into play in the following case:



Unfortunately, the “Stream Lookup” step can only read lookup information from one stream, so we need to redo the transformation like this:

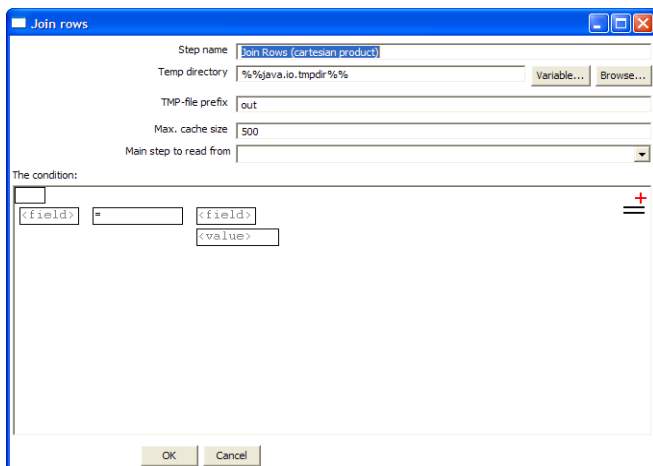


#### 8.5.13.4. Options

- ✓ Step name: Name of the step. This name has to be unique in a single transformation.

## 8.5.14. Join Rows (Cartesian product)

### 8.5.14.1. Screenshot

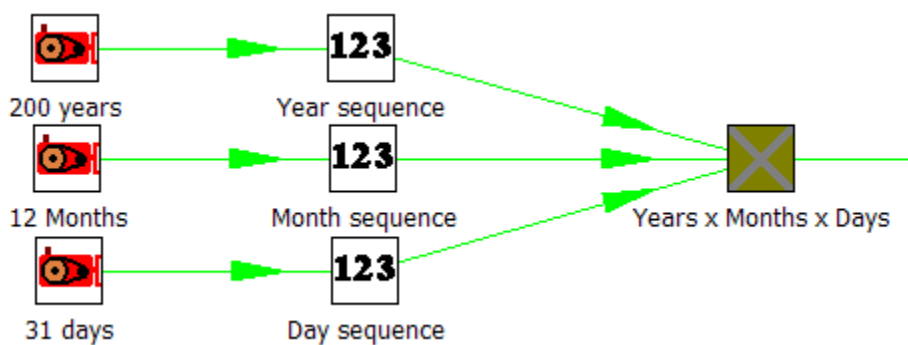


### 8.5.14.2. Icon



### 8.5.14.3. General description

This step allows you to produce combinations (Cartesian product) of all rows on the input streams. This is an example:



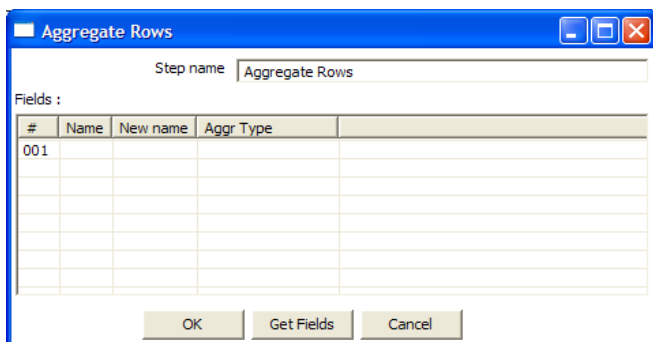
The “Years x Months x Days” step outputs all combinations of Year, Month and Day. (1900, 1, 1 → 2100, 12, 31) and can be used to create a date dimension.

#### 8.5.14.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Temp directory:** Specify the name of the directory where the system stores temporary files in case you want to combine more then the cached number of rows.
- ✓ **Temporary file prefix:** This is the prefix of the temporary files that will be generated.
- ✓ **Max. cache size:** The number of rows to cache before the systems reads data from temporary files. This is needed in case you want to combine large row sets that don't fit into memory.
- ✓ **Main step to read from:** Specifies the step to read the most data from. This step is not cached or spooled to disk, the others are.
- ✓ **The condition:** You can enter a complex condition to limit the number of output rows.

## 8.5.15. Aggregate Rows

### 8.5.15.1. Screenshot



### 8.5.15.2. Icon



### 8.5.15.3. General description

This step type allows you to quickly aggregate rows based upon all the rows.

These are the available aggregation types:

- SUM: the sum of a field
- AVERAGE: the average of a field
- COUNT: the number of (not null) values
- MIN: the minimum value of a field
- MAX: the maximum value of a field
- FIRST: the first value of a field
- LAST: the last value of a field

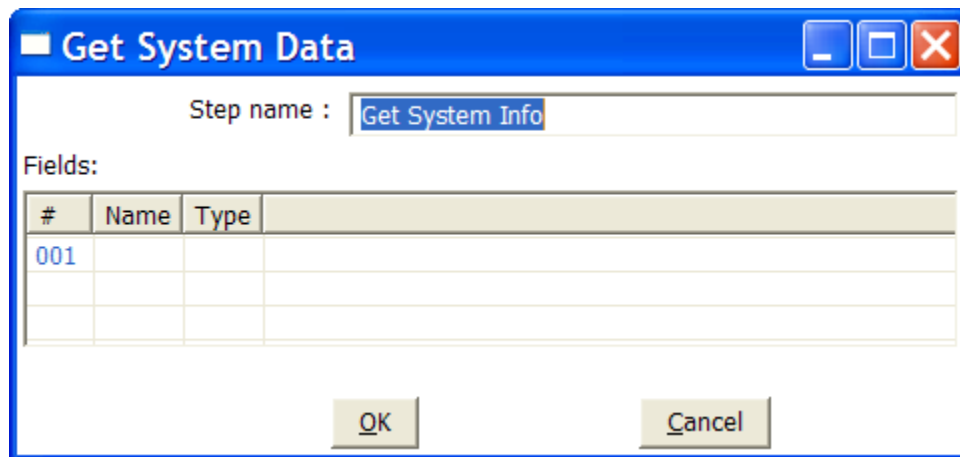
**NOTE:** This step type is deprecated. See the “Group By” step in [7.4.27. Group By](#) for a more powerful way of aggregating rows of data. The aggregate step will be removed in a future version.

### 8.5.15.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Fields:** The fields and the aggregation type.

## 8.5.16. Get System Info

### 8.5.16.1. Screenshot



### 8.5.16.2. Icon



### 8.5.16.3. General description

This step type retrieves information from the Kettle environment.  
 These are the possible pieces of information you can retrieve:

Option	Description
system date (variable)	System time, changes every time you ask a date.
system date (fixed)	System time, determined at the start of the transformation.
start date range	Start of date range, based upon information in ETL log table. See, also <a href="#">6. Transformation settings</a>
end date range	End of date range, based upon information in ETL log table. See, also <a href="#">6. Transformation settings</a>
Yesterday 00:00:00	Start of yesterday.
Yesterday 23:59:59	End of yesterday.
Today 00:00:00	Start of today.
Today 23:59:59	End of today.
Tomorrow 00:00:00	Start of tomorrow.
Tomorrow 23:59:59	End of tomorrow
First day of last month 00:00:00	Start of last month.
Last day of last month 23:59:59	End of last month.
First day of this month 00:00:00	Start of this month.
Last day of this month 23:59:59	End of this month.
First day of next month 00:00:00	Start of next month.
Last day of next month 23:59:59	End of next month.



Option	Description
copy of step	Copy nr of the step (see also: <a href="#">7.2. Launching several copies of a step</a> ).
transformation name	Name of the transformation.
transformation file name	File name of the transformation (XML only).
User that modified the transformation last	
Date when the transformation was modified last	
transformation batch ID	ID_BATCH value in the logging table, see <a href="#">6. Transformation settings</a> .
Hostname	Returns the hostname of the server.
IP address	Returns the IP address of the server.
command line argument 1	Argument 1 on the command line.
command line argument 2	Argument 2 on the command line.
command line argument 3	Argument 3 on the command line.
command line argument 4	Argument 4 on the command line.
command line argument 5	Argument 5 on the command line.
command line argument 6	Argument 6 on the command line.
command line argument 7	Argument 7 on the command line.
command line argument 8	Argument 8 on the command line.
command line argument 9	Argument 9 on the command line.
command line argument 10	Argument 10 on the command line.

#### 8.5.16.4. Options

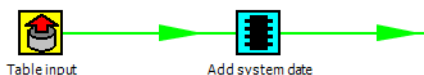
- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Fields:** The fields to output.

#### 8.5.16.5. Usage

The first type of usage is to simple get information from the system:

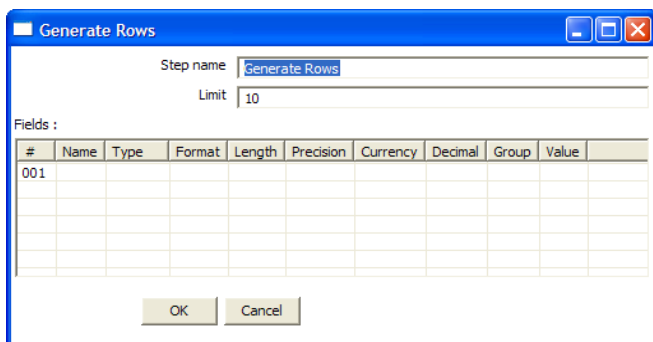


From version 2.3.0 on, this step also accepts input rows. The selected values will be added to the rows found in the input stream(s):



## 8.5.17. Generate Rows

### 8.5.17.1. Screenshot



#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value
001									

### 8.5.17.2. Icon



### 8.5.17.3. General description

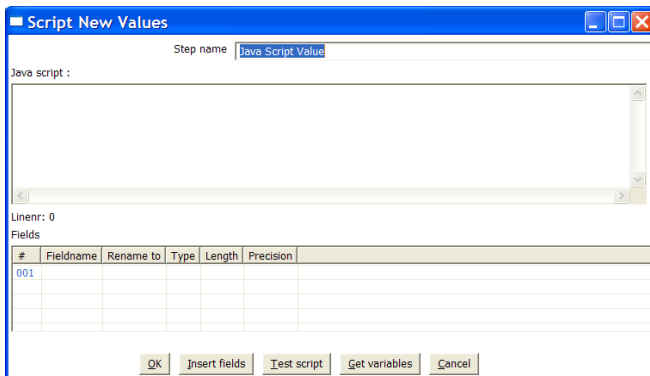
This step type outputs a number of rows, default empty but optionally containing a number of static fields.

### 8.5.17.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Limit:** The number of rows you want to output.
- ✓ **Fields:** Static fields you might want to include in the output row.

## 8.5.18. Java Script Value

### 8.5.18.1. Screenshot



### 8.5.18.2. Icon



### 8.5.18.3. General description

This step type allows you to do complex calculations using the JavaScript language.

### 8.5.18.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Java Script:** The script in question.
- ✓ **Fields:** These are the fields that we want to add to the output stream.

### 8.5.18.5. Extra

- ✓ **Insert fields:** Inserts the fields and the standard method to grab the value of the field.
- ✓ **Test script:** Tests whether or not the script compiles.
- ✓ **Get variables:** Gets the newly created variables and inserts them into the “Fields” grid.

### 8.5.18.6. Value functions

This is a list of functions that you can use to manipulate values...

- ✓ **Value Clone()**  
Builds a copy of a value and returns a Value.
- ✓ **void setName(String name)**  
Sets the name of a value.
- ✓ **String getName()**  
Get the name of a value.
- ✓ **void setValue(double num)**  
Set the value to a floating point value.

- ✓ `void setValue(String str)`  
Set the value to a floating point value.
- ✓ `void setValue(Date dat)`  
Set the value to a Date value.
- ✓ `void setValue(boolean bool)`  
Set the value to a Boolean value.
- ✓ `void setValue(long l)`  
Set the value to an integer value.
- ✓ `void setValue(Value v)`  
Set the value to the value contained in another field.
- ✓ `double getNumber()`  
Gets the value of a field as a floating point value.
- ✓ `String getString()`  
Gets the value of a field as a textual string representation.
- ✓ `int getStringLength()`  
Gets the length of the string representation.
- ✓ `Date getDate()`  
Gets the value of the field as a date value.
- ✓ `boolean getBoolean()`  
Gets the value of a field as a Boolean.  
**NOTE:** String "Y" or "true" is converted to true.  
**NOTE:** Numeric value 0 is converted to false, everything else is true.
- ✓ `long getInteger()`  
Gets the value of a field as an integer.  
**NOTE:** Date fields are converted to the number of milliseconds since January 1st 1970 00:00:00 GMT.
- ✓ `boolean isEmpty()`  
If the value has no type, this function returns true.
- ✓ `boolean isString()`  
If the value is of type String, this function returns true.
- ✓ `boolean isDate()`  
If the value is of type String, this function returns true.
- ✓ `boolean isNumber()`  
If the value is of type Number, this function returns true.
- ✓ `boolean isBoolean()`  
If the value is of type Boolean, this function returns true.

- ✓ **boolean isInteger()**  
If the value is of type Integer, this function returns true.
- ✓ **boolean isNumeric()**  
If the value is of type Number or Integer, this function returns true.
- ✓ **String toString()**  
Returns the textual representation of the value.
- ✓ **String toString(boolean pad)**  
Returns the textual representation of the value, padded to the length of the string if pad = true.
- ✓ **String toStringMeta()**  
Returns the meta-data information of the value as a string.
- ✓ **void setLength(int l)**  
Sets the length of the value.
- ✓ **void setLength(int l, int p)**  
Sets the length and precision of the value.
- ✓ **int getLength()**  
Returns the length of the value.
- ✓ **int getPrecision()**  
Returns the precision of the value.
- ✓ **void setPrecision(int p)**  
Sets the precision of a value.
- ✓ **String getTypeDesc()**  
Returns the description of a value as a string. (e.g. "String", "Number", "Date", "Integer", "Boolean").
- ✓ **void setNull()**  
Sets the value to Null.
- ✓ **void clearNull()**  
Removes the null setting.
- ✓ **void setNull(boolean n)**
- ✓ **boolean isNull()**  
Returns true if the value is null.
- ✓ **int compare(Value v)**  
Compares two values and returns 1 if the first value is larger than the second, -1 if it is smaller and 0 if they are equal.
- ✓ **boolean equals(Object v)**  
Compares two values and returns true if the two values have the same value.

- ✓ `int hashCode()`  
Returns a signed 64 values representing the value in the form of a hash code.
- ✓ `Value negate()`  
If the value is numeric, multiplies the value by -1, in all other cases it doesn't do anything.
- ✓ `Value and(Value v)`  
Calculates the bitwise AND of two integer values.
- ✓ `Value xor(Value v)`  
Calculates the bitwise XOR of two integer values.
- ✓ `Value or(Value v)`  
Calculates the bitwise OR of two integer values.
- ✓ `Value bool_and(Value v)`  
Calculates the boolean AND of two boolean values.
- ✓ `Value bool_or(Value v)`  
Calculates the boolean OR of two boolean values.
- ✓ `Value bool_xor(Value v)`  
Calculates the boolean XOR of two boolean values.
- ✓ `Value bool_not()`  
Calculates the boolean NOT of a boolean value.
- ✓ `Value greater_equal(Value v)`  
Compares two values and sets the first to true if the second is greater or equal to the first.
- ✓ `Value smaller_equal(Value v)`  
Compares two values and sets the first to true if the second is smaller or equal to the first.
- ✓ `Value different(Value v)`  
Compares two values and sets the first to true if the second is different from the first.
- ✓ `Value equal(Value v)`  
Compares two values and sets the first to true if the second is equal to the first.
- ✓ `Value like(Value v)`  
Sets the first value to true if the second string is part of the first.
- ✓ `Value greater(Value v)`  
Compares two values and sets the first to true if the second is greater than the first.
- ✓ `Value smaller(Value v)`  
Compares two values and sets the first to true if the second is smaller than the first.

- ✓ Value minus(double v)
- ✓ Value minus(long v)
- ✓ Value minus(int v)
- ✓ Value minus(byte v)
- ✓ Value minus(Value v)  
Subtracts v from the field value.
  
- ✓ Value plus(double v)
- ✓ Value plus(long v)
- ✓ Value plus(int v)
- ✓ Value plus(byte v)
- ✓ Value plus(Value v)  
Adds v to the field value.
  
- ✓ Value divide(double v)
- ✓ Value divide(long v)
- ✓ Value divide(int v)
- ✓ Value divide(byte v)
- ✓ Value divide(Value v)  
Divides the field value by v.
  
- ✓ Value multiply(double v)
- ✓ Value multiply(long v)
- ✓ Value multiply(int v)
- ✓ Value multiply(byte v)
- ✓ Value multiply(Value v)  
Multiplies the field value by v.  
**NOTE:** *Strings can be multiplied as well: the result is v times the string concatenated.*
  
- ✓ Value abs()  
Sets the field value to the –field value if the value was negative.
  
- ✓ Value acos()  
Sets the field value to the cosine of the number value.
  
- ✓ Value asin()  
Sets the field value to the arc sine of the number value.
  
- ✓ Value atan()  
Sets the field value to the arc tangents of the number value.
  
- ✓ Value atan2(Value arg0)
- ✓ Value atan2(double arg0)  
Sets the field value to the second arc tangents of the number value.

- ✓ Value ceil()  
Sets the field value to the ceiling of a number value.
- ✓ Value cos()  
Sets the field value to the cosine of a number value.
- ✓ Value cosh()  
Sets the field value to the hyperbolic cosine of a number value.
- ✓ Value exp()  
Sets the field value to the exp of a number value.
- ✓ Value floor()  
Sets the field value to the floor of a number value.
- ✓ Value initcap()  
Sets the all first characters of words in a string to uppercase.  
"matt casters" → "Matt Casters"
- ✓ Value length()  
Sets the value of the field to the length of the String value.
- ✓ Value log()  
Sets the field value to the log of a number value.
- ✓ Value lower()  
Sets the field value to the string value in lowercase.
- ✓ Value lpad(Value len)
- ✓ Value lpad(Value len, Value padstr)
- ✓ Value lpad(int len)
- ✓ Value lpad(int len, String padstr)  
Sets the field value to the string value, left padded to a certain length.  
Default the padding string is a single space.  
Optionally, you can specify your own padding string.
- ✓ Value ltrim()  
Sets the field value to the string, without spaces to the left of the string.
- ✓ Value mod(Value arg)
- ✓ Value mod(double arg0)  
Sets the value to the modulus of the first and the second number.
- ✓ Value nvl(Value alt)  
If the field value is Null, set the value to alt.
- ✓ Value power(Value arg)
- ✓ Value power(double arg0)  
Raises the field value to the power arg.



- ✓ Value replace(Value repl, Value with)
- ✓ Value replace(String repl, String with)  
Replaces a string in the field value with another.
- ✓ Value round()  
Rounds the field value to the nearest integer.
- ✓ Value rpad(Value len)
- ✓ Value rpad(Value len, Value padstr)
- ✓ Value rpad(int len)
- ✓ Value rpad(int len, String padstr)  
Sets the field value to the string value, right padded to a certain length.  
Default the padding string is a single space.  
Optionally, you can specify your own padding string.
- ✓ Value rtrim()  
Remove the spaces to the right of the field value.
- ✓ Value sign()  
Sets the value of the string to -1, 0 or 1 in case the field value is negative, zero or positive.
- ✓ Value sin()  
Sets the value of the field to the sine of the number value.
- ✓ Value sqrt()  
Sets the value of the field to the square root of the number value.
- ✓ Value substr(Value from, Value to)
- ✓ Value substr(Value from)
- ✓ Value substr(int from)
- ✓ Value substr(int from, int to)  
Sets the value of the field to the substring of the string value.
- ✓ Value sysdate()  
Sets the field value to the system date.
- ✓ Value tan(Value args[])  
Sets the field value to the tangents of the number value.

- ✓ Value num2str()
- ✓ Value num2str(String arg0)
- ✓ Value num2str(String arg0, String arg1)
- ✓ Value num2str(String arg0, String arg1, String arg2)
- ✓ Value num2str(String arg0, String arg1, String arg2, String arg3)

Converts a number to a string.

Arg0: format pattern, see also [7.4.1.5.1. Number formats](#)

Arg1: Decimal separator (either . or ,)

Arg2: Grouping separator (either . or ,)

Arg3: Currency symbol

For example converting value:

1234.56	using num2str("####,##0.00", ",", ".")	gives	1.234,56
.23	using num2str("####,##0.00", ",", ".")	gives	0,23
1234.56	using num2str("000,000.00", ",", ".")	gives	001.234,56

- ✓ Value dat2str()
- ✓ Value dat2str(String arg0)
- ✓ Value dat2str(String arg0, String arg1)
 

Converts a date into a string.

Arg0: format pattern, see also [7.4.1.5.1. Number formats](#)

Arg1: localized date-time pattern characters (u, t)
- ✓ Value num2dat()
 

Converts a number to a date based upon the number of milliseconds since January 1<sup>st</sup>, 1970 00:00:00 GMT.
- ✓ Value str2dat(String arg0)
- ✓ Value str2dat(String arg0, String arg1)
 

Converts a string to a date.

Arg0: format pattern, see also [7.4.1.5.1. Number formats](#)

Arg1: localized date-time pattern characters (u, t)
- ✓ Value str2num()
- ✓ Value str2num(String arg0)
- ✓ Value str2num(String arg0, String arg1)
- ✓ Value str2num(String arg0, String arg1, String arg2)
- ✓ Value str2num(String arg0, String arg1, String arg2, String arg3)
 

Converts a string into a number.

Arg0: format pattern, see also [7.4.1.5.1. Number formats](#)

Arg1: Decimal separator (either . or ,)

Arg2: Grouping separator (either . or ,)

Arg3: Currency symbol
- ✓ Value dat2num()
 

Converts a date into a number being the number of milliseconds since January 1<sup>st</sup>, 1970 00:00:00 GMT.
- ✓ Value trim()
 

Remove spaces left and right of the string value.

- ✓ Value upper()  
Sets the field value to the uppercase string value.
- ✓ Value e()  
Sets the value to e
- ✓ Value pi()  
Sets the value to  $\pi$
- ✓ Value add\_months(int months)  
Adds a number of months to the date value.
- ✓ Value last\_day()  
Sets the field value to the last day of the month of the date value.
- ✓ Value first\_day()  
Sets the field value to the first day of the month of the date value.
- ✓ Value trunc()
- ✓ Value trunc(double level)
- ✓ Value trunc(int level)  
Set the field value to the truncated number or date value.  
Level means the number of positions behind the comma or in the case of a date, 5=months, 4=days, 3=hours, 2=minutes, 1=seconds
- ✓ Value hexEncode()  
Encode a String value in its hexadecimal representation. E.g. If value is a string "a", the result would be "61".
- ✓ Value hexDecode()  
Decode a String value from its hexadecimal representation. E.g. If value is a string "61", the result would be "a". If the input string value is odd a leading 0 will be silently added.

### 8.5.18.7. JavaScript Examples

#### 8.5.18.7.1. Remember the previous row:

Sometimes it can be useful to know the value of the previous row. This can be accomplished by this piece of code:

```
var prev_row;
if (prev_row == null) prev_row = row;

...
String previousName = prev_row.getString("Name", "-");
...

prev_row = row;
```

row is a special field that contains all values in the current row.

#### 8.5.18.8. Sets the location name of an address to uppercase

```
location.upper();
```

#### 8.5.18.9. Extract information from a date field

```
// Year/Month/Day representation:
ymd = date_field.Clone().dat2str("yyyy/MM/dd").getString();

// Day/Month/Year representation:
dmy = date_field.Clone().dat2str("dd/MM/yyyy").getString();

// Year/Month :
ym = date_field.Clone().dat2str("yyyy/MM").getString();

// Long description of the month in the local language:
month_long_desc= date_field.Clone().dat2str("MMMM").initcap().getString();

// Week of the year (1-53)
week_of_year = date_field.Clone().dat2str("w").getInteger();

// day of week, short description (MON-SUN)
day_of_week_short_desc =date_field.Clone().dat2str("EEE").upper().getString();

// Day of the week (Monday-Sunday)
day_of_week_desc = date_field.Clone().dat2str("EEEE").initcap().getString();

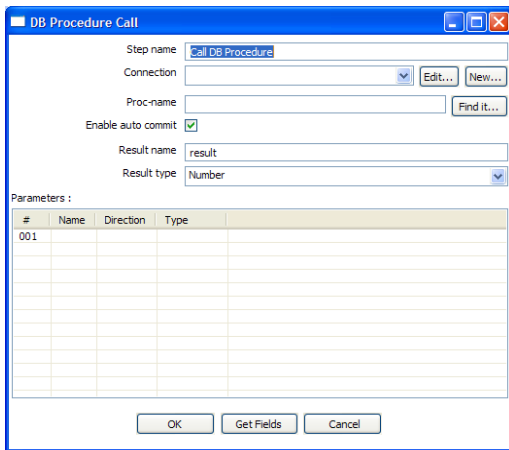
// Day of week (1-7)
day_of_week = date_field.Clone().dat2str("F").getInteger();
```

See also: [7.4.1.5.1. Number formats](#)

**NOTE:** If you don't use Clone(), the original value will be overwritten by the methods that work on the Kettle Values.

## 8.5.19. Call DB Procedure

### 8.5.19.1. Screenshot



### 8.5.19.2. Icon



### 8.5.19.3. General description

This step type allows you to execute a database procedure (or function) and get the result(s) back.

### 8.5.19.4. Options

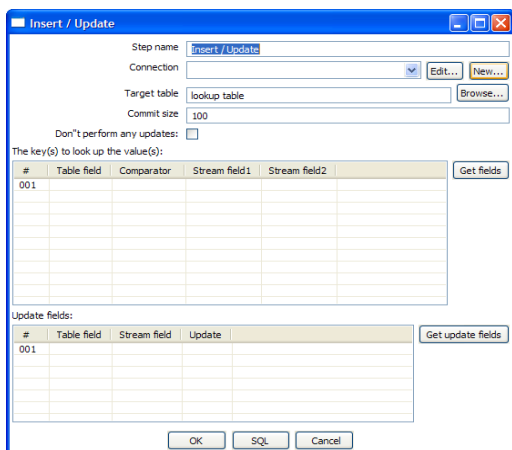
- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Connection:** Name of the database connection on which the procedure resides.
- ✓ **Proc-name:** Name of the procedure or function to call.
- ✓ **Enable auto-commit:** In some situations you want to do updates in the database using the specified procedure. In that case you can either have the changes done using auto-commit or by disabling this. If auto-commit is disabled, a single commit is being performed after the last row was received by this step.
- ✓ **Result name:** Name of the result of the function call, leave this empty in case of procedure.
- ✓ **Result type:** Type of the result of the function call. Not used in case of a procedure.
- ✓ **Parameters:** List of parameters that the procedure or function needs
  - Field name: Name of the field.
  - Direction: Can be either IN (input only), OUT (output only), INOUT (value is changed on the database).
  - Type: Used for output parameters so that Kettle knows what comes back.

### 8.5.19.5. Extra

- ✓ **“Find it...” button:** searches on the specified database connection for available procedures and functions (at the moment only on Oracle and SQLServer).
- ✓ **“Get fields” button:** this function fills in all the fields in the input streams to make your life easier. Simply delete the lines you don’t need and re-order the ones you do need.

## 8.5.20. Insert / Update

### 8.5.20.1. Screenshot



### 8.5.20.2. Icon



### 8.5.20.3. General description

This step type first looks up a row in a table using one or more lookup keys. If the row can't be found, it inserts the row. If it can be found and the fields to update are the same, nothing is done. If they are not all the same, the row in the table is updated.

### 8.5.20.4. Options

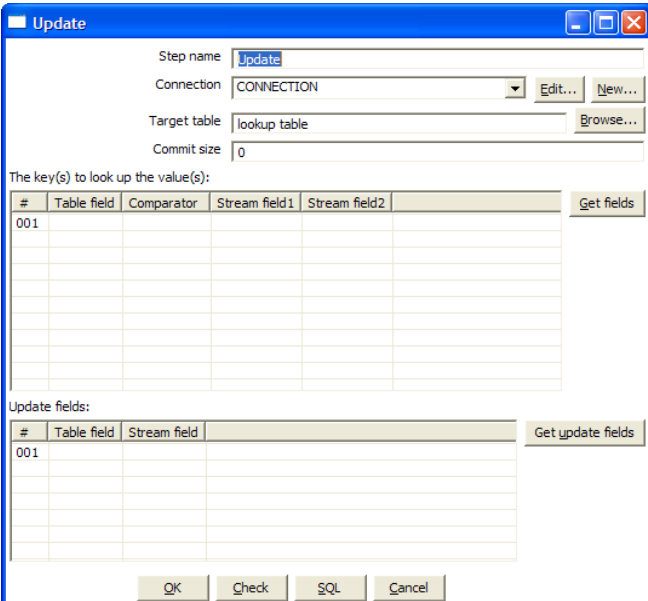
- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Connection:** Name of the database connection on which the target table resides.
- ✓ **Target table:** Name of the table in which you want to do the insert or update.
- ✓ **Commit size:** The number of rows to change (insert / update) before running a commit.
- ✓ **Don't perform any updates:** If this option is checked, the values in the database are never updated. Only inserts are done.
- ✓ **Keys:** here you can specify a list of field values and comparators.  
You can use the following comparators: =, <>, <, <=, >, >=, LIKE, BETWEEN, IS NULL, IS NOT NULL
- ✓ **Update fields:** Specify all fields in the table you want to insert/update *including the keys*. Please note that you can avoid updates on certain fields by specifying N in the update column.

#### 8.5.20.5. Extra

- ✓ **“Get fields” button**: get the fields from the input stream(s) and fills them in into the keys grid.
- ✓ **“Get update fields” button**: gets the update fields from the input streams and fills them in into the update grid.
- ✓ **“Check” button**: checks whether or not all fields are available in the target table.
- ✓ **“SQL” button**: generates the SQL to create the table and indexes for correct operation.

### 8.5.21. Update

#### 8.5.21.1. Screenshot



#### 8.5.21.2. Icon

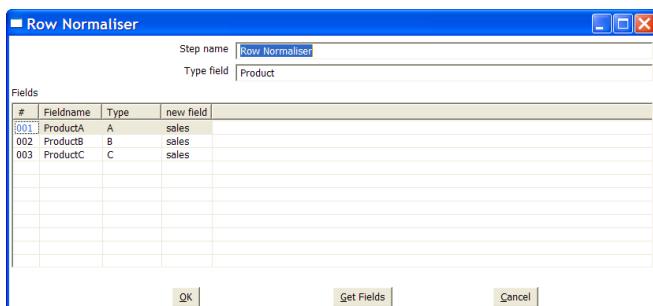


#### 8.5.21.3. General description

This step is the same as the [8.4.20. Insert / Update](#) step, except that no insert is ever done in the database table. ONLY updates are performed.

## 8.5.22. Row Normaliser

### 8.5.22.1. Screenshot



### 8.5.22.2. Icon



### 8.5.22.3. General description

This step normalizes data back from pivoted tables for example. Take this example of product sales data:

Month	Product A	Product B	Product C
2003/01	10	5	17
2003/02	12	7	19
...	...	...	...



If you want to convert this data into:

Month	Product	sales
2003/01	A	10
2003/01	B	5
2003/01	C	17
2003/02	A	12
2003/02	B	7
2003/02	C	19
...	...	...

For example if you want to update a fact table, this form of data is a lot easier to handle. Well, that's what this step type does.

#### 8.5.22.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Type field:** The name of the type field. (Product in our example)
- ✓ **Fields:** A list of fields to normalize:
  - **Fieldname:** Name of the fields to normalize (Product A → C in our example).
  - **Type:** Give a string to classify the field (A, B or C in our example).
  - **New field:** You can give one or more fields where the new value should transferred to (sales in our example).

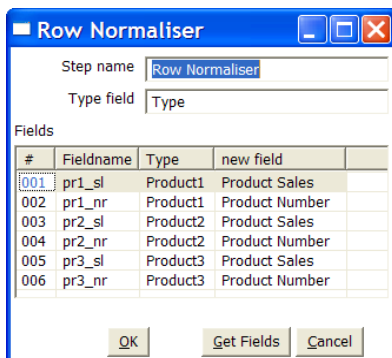
**NOTE:** *It is possible to normalize more then on field at a time.  
Consider this example:*

DATE	PR1 NR	PR1 SL	PR2 NR	PR2 SL	PR3 NR	PR3 SL
20030101	5	100	10	250	4	150
...	...	...	...	...	...	...

*You can convert this into:*

DATE	Type	Product Sales	Product Number
20030101	Product1	100	5
20030101	Product2	250	10
20030101	Product3	150	4
...	...	...	...

*This would be the setup to do it with:*



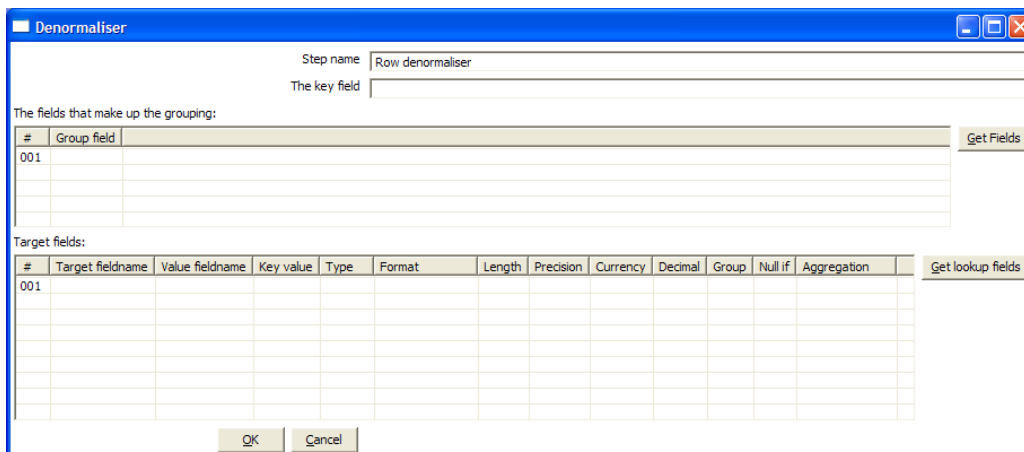
#	Fieldname	Type	new field
001	pr1_sl	Product1	Product Sales
002	pr1_nr	Product1	Product Number
003	pr2_sl	Product2	Product Sales
004	pr2_nr	Product2	Product Number
005	pr3_sl	Product3	Product Sales
006	pr3_nr	Product3	Product Number

#### 8.5.22.5. Extra

- ✓ **“Get fields” button:** this fills in all fields that are available on the input stream(s).

## 8.5.23. Denormaliser

### 8.5.23.1. Screenshot



### 8.5.23.2. Icon



### 8.5.23.3. General description

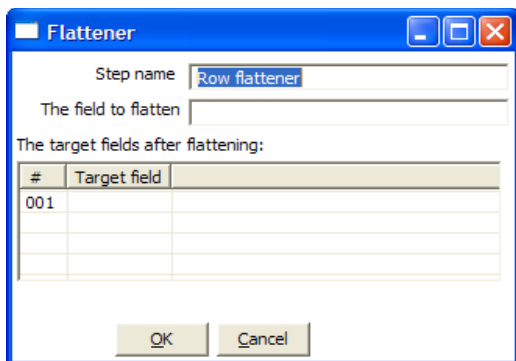
This step allows you de-normalise data by looking up key-value pairs. It also allows you to immediately convert data types.

### 8.5.23.4. Options

Option	Description
Step name	Name of the step. This name has to be unique in a single transformation.
Key field	The field that defined the key.
Group fields	Specify the fields that make up the grouping here.
Target fields	<p>Specify the fields to de-normalise. You do it by specifying the String value for the key field (see above).</p> <p>Options are provided to convert data types. Mostly people use Strings as key-value pairs so you often need to convert to Integer, Number or Date.</p> <p>In case you get key-value pair collisions (key is not unique for the group specified) you can specify the aggregation method to use.</p>

## 8.5.24. Flattener

### 8.5.24.1. Screenshot



### 8.5.24.2. Icon



### 8.5.24.3. General description

This step allows you flatten sequentially provided data.

### 8.5.24.4. Options

<i>Option</i>	<i>Description</i>
Step name	Name of the step. This name has to be unique in a single transformation.
The field to flatten	The field that needs to be flattened into different target fields.
Target fields	The name of the target field to flatten to.

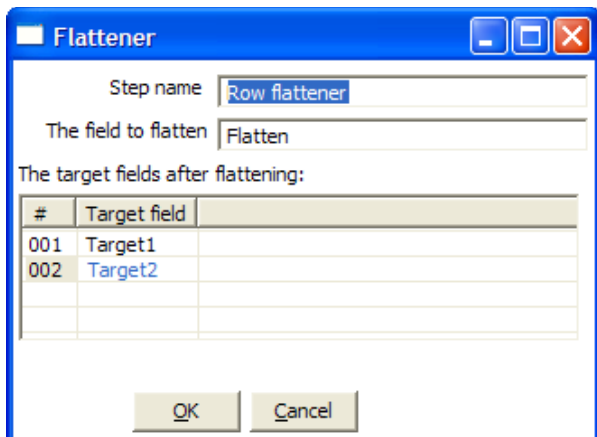
### 8.5.24.5. Example

<i>Field1</i>	<i>Field2</i>	<i>Field3</i>	<i>Flatten</i>
A	B	C	One
A	B	C	Two
D	E	F	Three
D	E	F	Four

Can be flattened to:

<i>Field1</i>	<i>Field2</i>	<i>Field3</i>	<i>Target1</i>	<i>Target2</i>
A	B	C	One	Two
D	E	F	Three	Four

In the example above, this is what the dialog looks like:



Step name: Row flattener

The field to flatten: Flatten

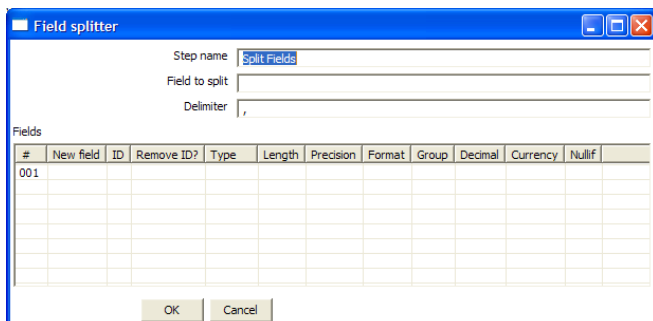
The target fields after flattening:

#	Target field
001	Target1
002	Target2

OK Cancel

## 8.5.25. Split Fields

### 8.5.25.1. Screenshot



### 8.5.25.2. Icon



### 8.5.25.3. General description

This step allows you to split fields based upon delimiter information.

### 8.5.25.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Field to split:** The name of the field you want to split.
- ✓ **Delimiter:** Delimiter that determines the end of a field.
- ✓ **Fields:** List of fields to split into.

**Example:** SALES\_VALUES field containing: “500,300,200,100”

Use these settings to split the field into 4 new fields:

- Delimiter: ,
- Field: SALES1, SALES2, SALES3, SALES4
- Id:
- remove ID no, no, no, no
- type: Number, Number, Number, Number
- format: ###.##, ###.##, ###.##, ###.##
- group:
- decimal: .
- currency:
- length: 3, 3, 3, 3
- precision: 0, 0, 0, 0

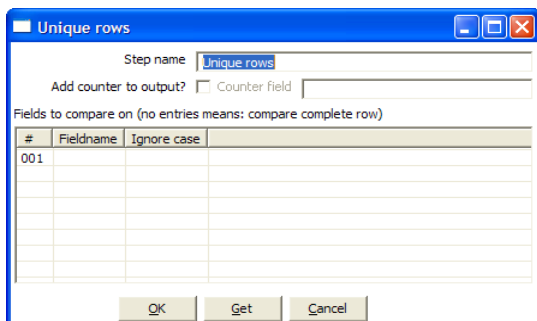
**Example:** SALES\_VALUES field containing “Sales2=310.50, Sales4=150.23”

Use these settings to split the field into 4 new fields:

- Delimiter: ,
- Field: SALES1, SALES2, SALES3, SALES4
- Id: Sales1=, Sales2=, Sales3=, Sales4=
- remove ID yes, yes, yes, yes
- type: Number, Number, Number, Number
- format: ###.##, ###.##, ###.##, ###.##
- group:
- decimal: .
- currency:
- length: 7, 7, 7, 7
- precision: 2, 2, 2, 2

## 8.5.26. Unique rows

### 8.5.26.1. Screenshot



### 8.5.26.2. Icon



### 8.5.26.3. General description

This step removes double rows from the input stream(s).

**IMPORTANT:** Make sure that the input stream is sorted! Otherwise only consecutive double rows are evaluated correctly. (sometimes this can be the whole idea but just be careful :-))

### 8.5.26.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Add counter to output?:** Enable this if you want to know how many rows were doubles for each row in the output.
- ✓ **Counter fields:** Name of the numeric field containing the number of rows
- ✓ **Fieldnames:** a list of field names on which the uniqueness is compared. Data in the other fields of the row is ignored.
- ✓ A flag "Ignore case" allowing you to indicate whether or not the uniqueness has to be verified by comparing (in the case of String fields) case sensitive or not.  
For example: Kettle, KETTLE, kettle all will be the same in case the compare is done case insensitive. In this case, the first occurrence (Kettle) will be passed to the next step(s).

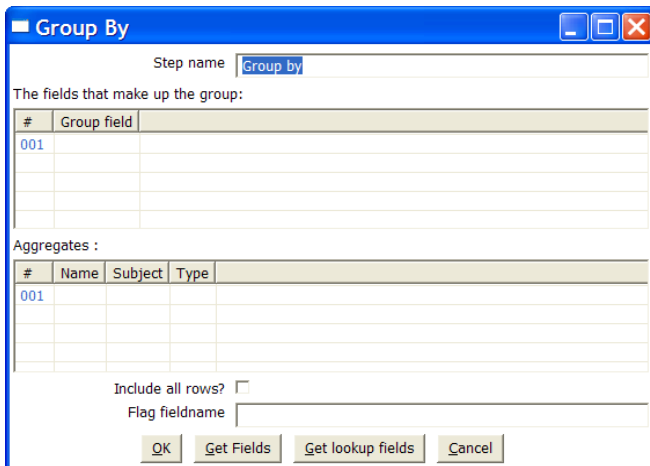
### 8.5.26.5. Extra

- ✓ "Get" button to automatically fill in the fields that are available to the step.



## 8.5.27. Group By

### 8.5.27.1. Screenshot



### 8.5.27.2. Icon



### 8.5.27.3. General description

This step allows you to calculate values over a defined group of fields.

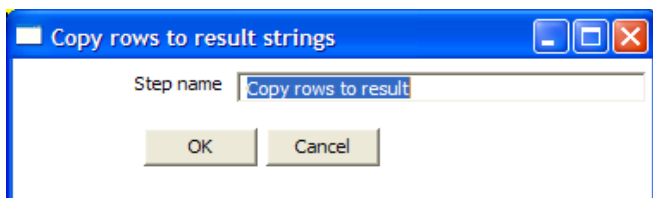
Examples that are common are: calculate the average sales per product, get the number of yellow shirts that we have in stock, etc.

### 8.5.27.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **Group fields:** Specify the fields over which you want to group.
- ✓ **Aggregates:** Specify the fields that need to be aggregated, the method and the name of the resulting new field.
- ✓ **Include all rows:** Check this if you want all rows in the output, not just the aggregation. To differentiate between the 2 types of rows in the output, we need a flag in the output. You need to specify the name of the flag field in that case. (the type is boolean)

## 8.5.28. Copy rows to result

### 8.5.28.1. Screenshot



### 8.5.28.2. Icon

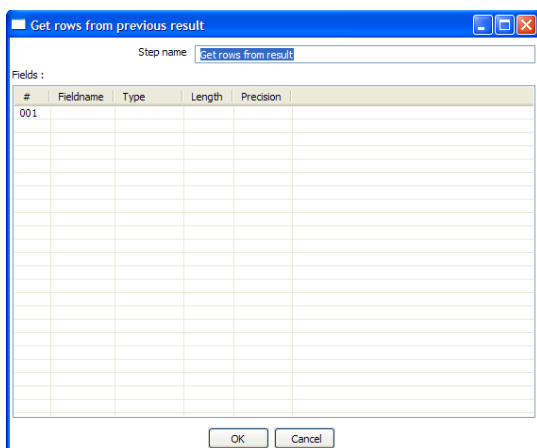


### 8.5.28.3. General description

This step allows you to transfer rows of data (in memory) to the **next** transformation (job entry) in a job. (see also: Chef user documentation)

## 8.5.29. Get rows from result

### 8.5.29.1. Screenshot



### 8.5.29.2. Icon

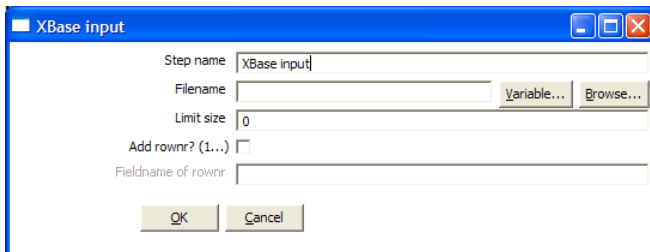


### 8.5.29.3. General description

This step returns rows that were previously generated by **another** transformation in a job. The rows were passed on to this step using the “Copy rows to result” step. To allow you to design more easily, you can enter the meta-data of the fields you're expecting from the previous transformation in a job.

## 8.5.30. XBase input

### 8.5.30.1. Screenshot



### 8.5.30.2. Icon



### 8.5.30.3. General description

With this step it is possible to read data from most types of DBF file derivatives called the XBase family. (dBase III/IV, Foxpro, Clipper, ...)

### 8.5.30.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ **The filename:** with variable support.
- ✓ **Limit size:** Only read this number of rows, 0 means: unlimited.
- ✓ **Add rownr?:** This option adds a field to the output with the specified name that contains the row number.



### 8.5.31.2. Icon



### 8.5.31.3. General description

With this step you can read data from one or more Excel files. It works on any system on which Kettle is supported.

### 8.5.31.4. Options

- ✓ Step name: Name of the step. This name has to be unique in a single transformation.

#### **File tab:**

- ✓ The filenames, with variable support.
- ✓ Please also see: [8.4.1.4.1. Filename specification options](#) and [8.4.1.4.2. Accept filenames from previous step](#)

#### **Sheets tab:**

- ✓ The sheets to import. *Use the “Get sheetname(s)” button to fill in the available sheets automatically.* Please note that you need to specify the start row and column, the coordinate where the step should start reading.

#### **Content tab:**

- ✓ **Header:** Check if the sheets specified have a header row that we need to skip.
- ✓ **No empty rows:** Check this if you don't want empty rows in the output of this step.
- ✓ **Stop on empty rows:** This will make the step stop reading the current sheet of a file when a empty line is encountered.
- ✓ **Filename field:** Specify a field name to include the filename in the output of this step.
- ✓ **Sheetname field:** Specify a field name to include the sheetname in the output of this step.
- ✓ **Row number field:** Specify a field name to include the row number in the output of the step.
- ✓ **Limit:** limit the number of rows to this number, 0 means: all rows.

#### **Error handling tab:**

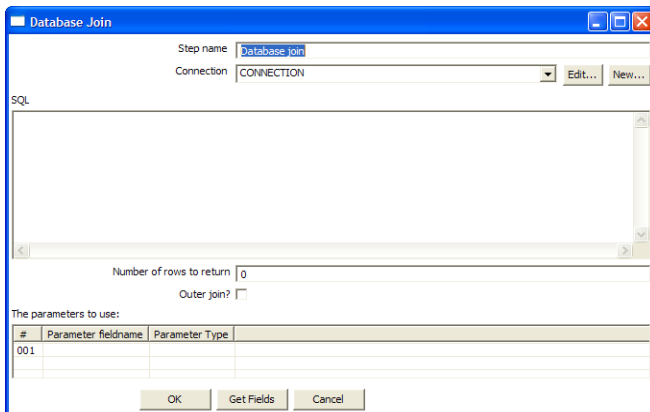
- ✓ Please see [8.4.1.4.4. Error handling](#) for a detailed description on error handling.

#### **Fields tab:**

- ✓ Here you can specify the fields that need to be read from the Excel files. A button “Get fields from header row” is provided to automatically fill in the available fields **if** the sheets have a header row.
- ✓ To allow for conversion logic, there is now support for the specification of data types. For example if you want to read a Date and you have a String value in the Excel file, you can specify the conversion mask. **Note:** in the case of Number to Date conversion (example: 20051028 --> October 28<sup>th</sup>, 2005) you should simply specify the conversion mask yyyyMMdd because there will be an implicit Number to String conversion taking place before doing the String to Date conversion.

## 8.5.32. Database Join

### 8.5.32.1. Screenshot



### 8.5.32.2. Icon



### 8.5.32.3. General description

Using data from previous steps, this step allows you to run a query against a database. The parameters for this query can be specified:

- as question marks (?) in the SQL query.
- as fields in the data grid.

The two need to be in the same order.

For example, this step allows you to run queries looking up the oldest person that bought a certain product like this:

```
SELECT  customernr
FROM    product_orders, customer
WHERE   orders.customernr = customer.customernr
AND     orders.productnr = ?
ORDER BY customer.date_of_birth
```

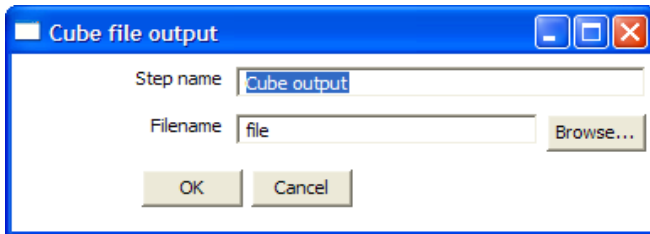
All you then need to specify as a parameter is the productnr and you'll get the customernr included in the result.

### 8.5.32.4. Options

- ✓ **Step name:** Name of the step. This name has to be unique in a single transformation.
- ✓ The database connection to use.
- ✓ **SQL:** The SQL query to launch towards the database. (use question marks (?) as parameter placeholders.
- ✓ **Number of rows to return:** 0 means all, any other number limits the number of rows.
- ✓ **Outer join?:** Check this to always return a result, even if the query didn't return a result.
- ✓ The parameters to use in the query.

### 8.5.33. Cube output

#### 8.5.33.1. Screenshot



#### 8.5.33.2. Icon



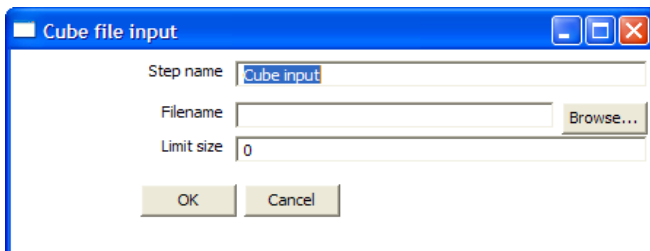
#### 8.5.33.3. General description

This step stores rows of data in a binary form in a file. It has the advantage over a text (flat) file that the content doesn't have to be parsed when read back. This is because the meta-data is stored in the cube file as well.

**NOTE:** This step should probably not be used as the tool that could use these Cube files was never created, the cube files are not using any “standard” format.

### 8.5.34. Cube input

#### 8.5.34.1. Screenshot



#### 8.5.34.2. Icon



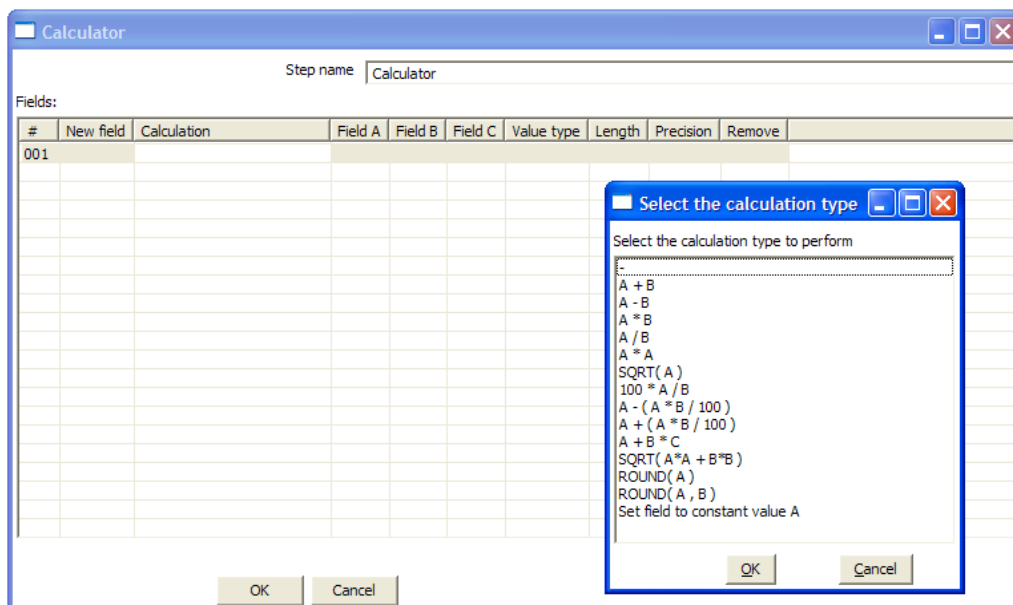
#### 8.5.34.3. General description

Read rows of data from a binary Kettle cube file. (see also “Cube output”)

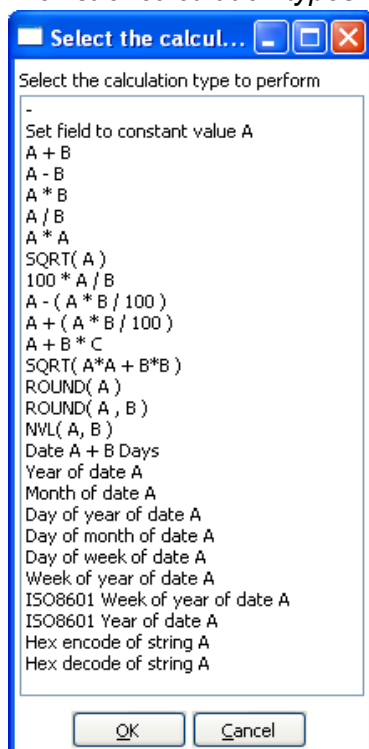
**NOTE:** This step should probably not be used as the tool that could use these Cube files was never created, the cube files are not using any “standard” format.

## 8.5.35. Calculator

### 8.5.35.1. Screenshot



The list of calculation types:





### 8.5.35.2. Icon



### 8.5.35.3. General description

This calculator step is making your life easier by providing a list of functions that can be executed on field values. Version 2.1 of Spoon is the first version to include this (short) list of functions. If you have a need for other (generic, often used) functions, simply write an e-mail to [requests@kettle.be](mailto:requests@kettle.be) specifying the needed functionality.

An important advantage Calculator has over custom JavaScript scripts is that the execution speed of Calculator is many times that of a script.

Besides the arguments (Field A, Field B and Field C) you also need to specify the return type of the function. You can also opt to remove the field from the result (output) after all values were calculated. This is useful for removing temporary values.

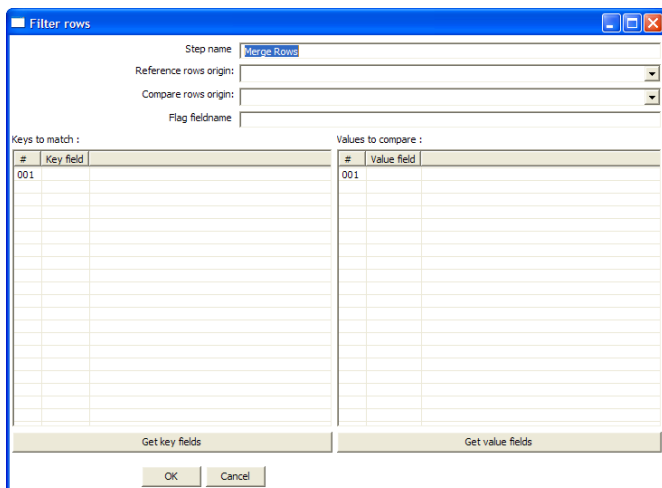
This is the current list of functions:

<i><b>Function</b></i>	<i><b>Description</b></i>	<i><b>Required fields</b></i>
Set field to constant A	Create a field with a constant value	A
A + B	A plus B	A and B
A - B	A minus B	A and B
A * B	A multiplied by B	A and B
A / B	A divided by B	A and B
A * A	The square of A	A
SQRT( A )	The square root of A	A
100 * A / B	Percentage of A in B	A and B
A - ( A * B / 100 )	Subtract B% of A	A and B
A + ( A * B / 100 )	Add B% to A	A and B
A + B * C	Add A and B times C	A, B and C
SQRT( A*A + B*B )	Calculate $\sqrt{(A^2+B^2)}$	A and B
ROUND( A )	Round A to the nearest integer	A
ROUND( A, B )	Round A to B decimal positions	A and B
Set field to constant A	Create a field with a constant value	A
Date A + B days	Add B days to Date field A	A and B
Year of date A	Calculate the year of date A	A
Month of date A	Calculate number the month of date A	A
Day of year of date A	Calculate the day of year (1-365)	A
Day of month of date A	Calculate the day of month (1-31)	A

<i><b>Function</b></i>	<i><b>Description</b></i>	<i><b>Required fields</b></i>
Day of week of date A	Calculate the day of week (1-7)	A
Week of year of date A	Calculate the week of year (1-54)	A
ISO8601 Week of year of date A	Calculate the week of the year ISO8601 style (1-53)	A
ISO8601 Year of date A	Calculate the year ISO8601 style	A
Hex encode of string A	Encode a string in its own hexadecimal representation	A
Hex decode of string A	Decode a string from its hexadecimal representation (add a leading 0 when A is of odd length)	A

## 8.5.36. Merge rows

### 8.5.36.1. Screenshot



### 8.5.36.2. Icon



### 8.5.36.3. General description

This step simply allows you to compare 2 streams of rows. This is useful if you want compare data from 2 different times. It is often used in situations where the source system of a data warehouse doesn't contain a data of last update.

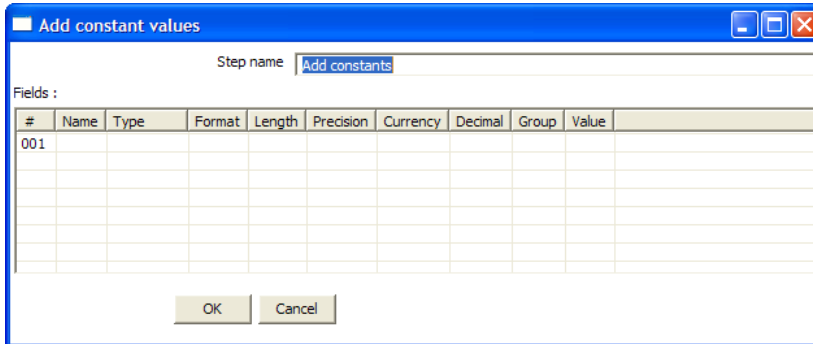
The 2 streams of rows, a reference stream and a compare stream, are merged. Each time only the last version of a row is passed onto the next steps. The row is marked as follows:

- ◆ “identical” : The key was found in both streams and the values to compare were identical.
- ◆ “changed” : The key was found in both streams but one or more values is different.
- ◆ “new” : The key was not found in the reference stream.
- ◆ “deleted” : The key was not found in the compare stream.

**IMPORTANT:** both streams need to be sorted on the specified key(s).

## 8.5.37. Add constants

### 8.5.37.1. Screenshot



### 8.5.37.2. Icon

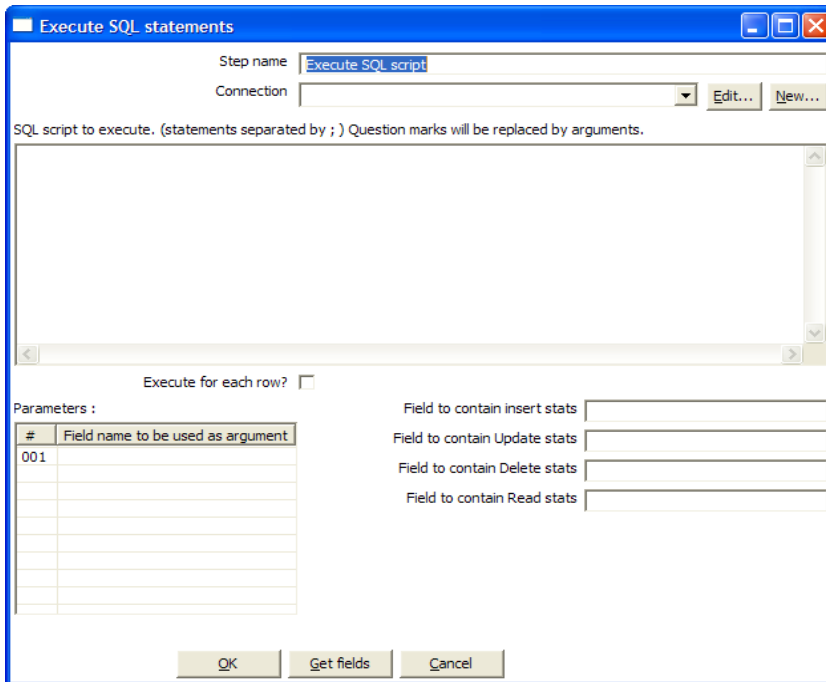


### 8.5.37.3. General description and usage

This step is the easiest and most performing way to add constants to a stream. The use is very simple: specify the name, type and value in the form of a string. Then specify the formats to convert the value into the chosen data type.

## 8.5.38. Execute SQL script

### 8.5.38.1. Screenshot



### 8.5.38.2. Icon

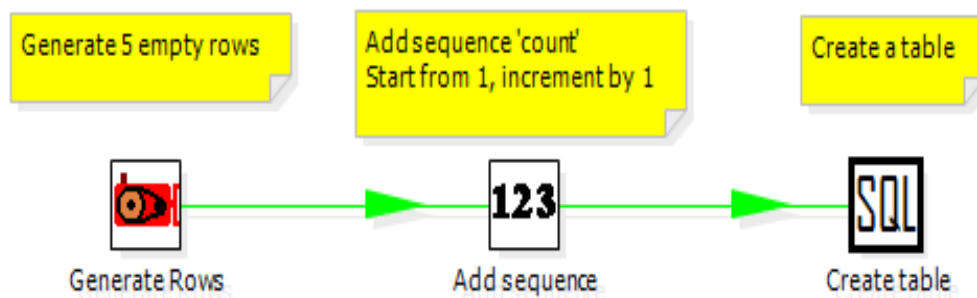


### 8.5.38.3. General description

You can execute SQL scripts with this step, either once, during the initialisation phase of the transformation, or once for every input-row that the step is given.

The second option can be used to use parameters in SQL scripts.

For example if you want to create 5 tables (tab1, tab2, tab3, tab4 and tab5) you could make such a transformation:



The SQL script to execute might look like this:

```
CREATE TABLE tab?
(
  a INTEGER
)
;
```

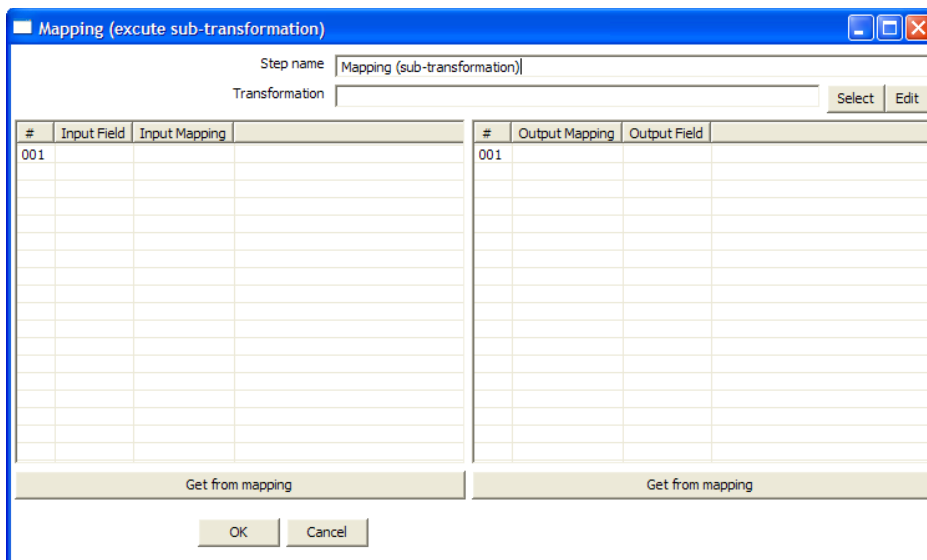
The field name to specify as parameter is then the “count” sequence we defined in the second step.

**NOTE:** *The execution of the transformation will halt when a statement in the script fails.*

As extra option, you can return the total number of inserts (INSERT INTO statements), updates (UPDATE table), deletes (DELETE FROM table) and reads (SELECT statements) by specifying the field names in the lower right of the dialog.

## 8.5.39. Mapping

### 8.5.39.1. Screenshot



### 8.5.39.2. Icon



### 8.5.39.3. General description & use

When you need to do a certain transformation over and over again, you can turn the repetitive part into a mapping. A mapping is a transformation that

- specifies how the input is going to be using a MappingInput step
- specified how the input fields are transformed: the field that are added and deleted

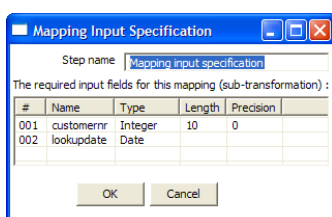
For example you can create mappings for dimension lookups so that you don't need to enter the natural keys etc. every time again.

In the dialog, you can specify the transformation name and even launch another Spoon window to edit the selected transformation.

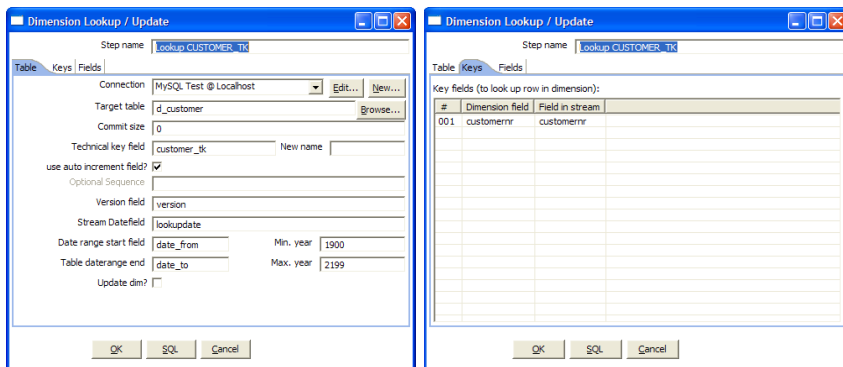
#### Example:

Suppose we want to create a mappings that does a lookup in the customer slowly changing dimension. In a larger warehouse, you need to specify the details for the dimension in question every time again. To get better re-use we want to create a mapping.

The details needed for the dimension lookup are in this case the customer number and the lookup reference date. These 2 input we specify in the mapping input step:

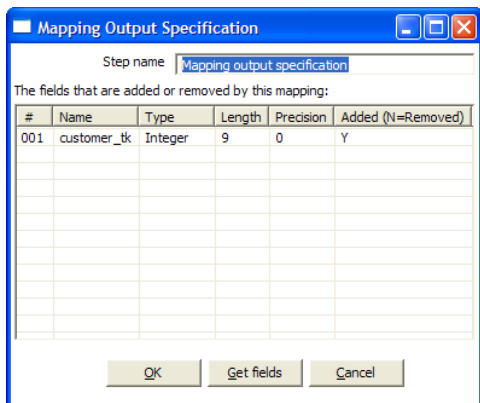


After this we can perform any calculation in our reusable transformation (Mapping), in our case we do a lookup in the dimension:





This dimension lookup step adds one field to the equation: customer\_tk.  
 We can specify the fields that were in the Mapping Output step:



Step name: Mapping output specification

The fields that are added or removed by this mapping:

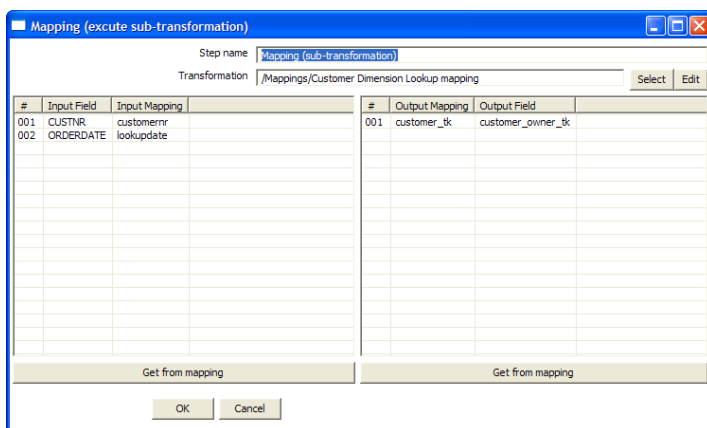
#	Name	Type	Length	Precision	Added (N=Removed)
001	customer_tk	Integer	9	0	Y

OK Get fields Cancel

The complete mapping looks like this:



When we want to re-use this mapping, this is how we can do it:



Step name: Mapping (sub-transformation)

Transformation: /Mappings/Custom Dimension Lookup mapping

#	Input Field	Input Mapping	#	Output Mapping	Output Field
001	CUSTNR	customernr	001	customer_tk	customer_owner_tk
002	ORDERDATE	lookupdate			

Get from mapping Get from mapping

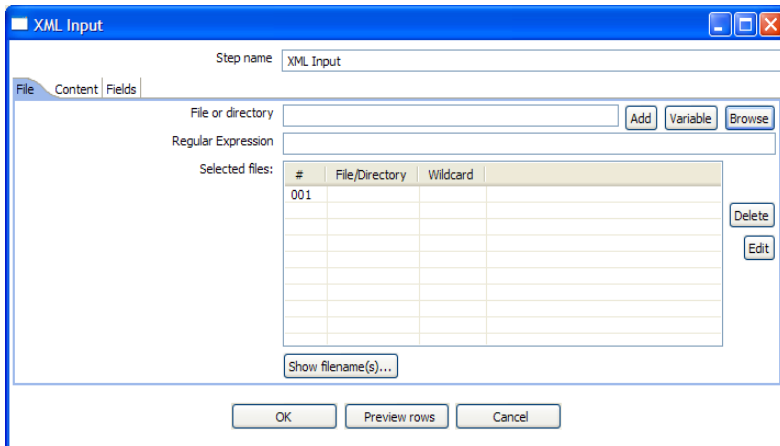
OK Cancel

As you can see, the way we do it is by “mapping” the stream fields of our choice to the required input fields of the mapping. Any added fields we can re-name on the output side.

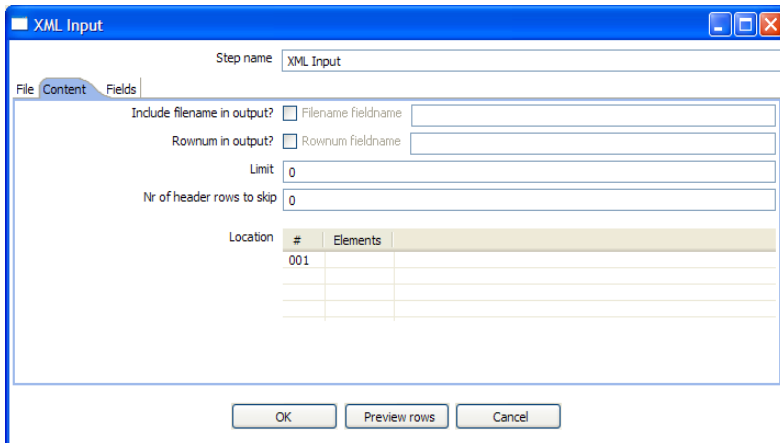
## 8.5.40. XML Input

### 8.5.40.1. Screenshot

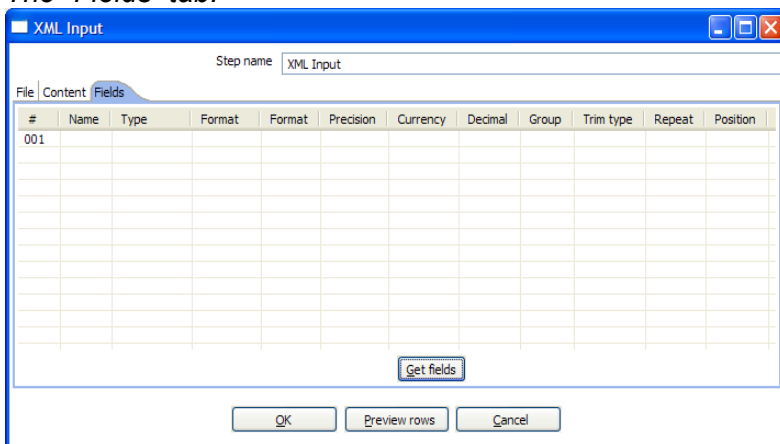
The “files” tab:



The “Content” tab:



The “Fields” tab:



#### 8.5.40.2. Icon



#### 8.5.40.3. General description

This step allows you to read information stored in XML files.

It does so by giving you an interface to define the filenames you want to read, the repeating part of the data part of the XML file and the fields to retrieve.

You specify the fields by the path to the Element or Attribute and by entering conversion masks, data types and other meta-data.

#### 8.5.40.4. Options

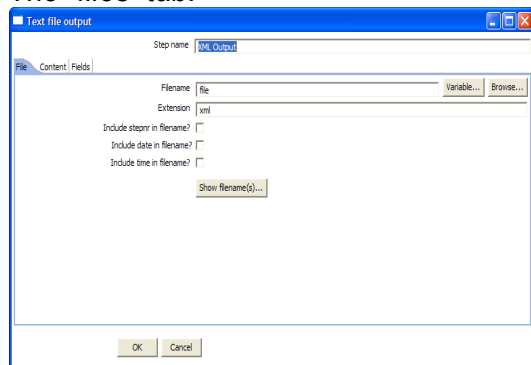
Option	Description
Step name	Name of the step. This name has to be unique in a single transformation.
File or directory	This field specifies the location and/or name of the input text file. <b>NOTE:</b> <i>press the “add” button to add the file/directory/wildcard combination to the list of selected files (grid) below.</i>
Regular expression	Specify the regular expression you want to use to select the files in the directory specified in the previous option.
Include filename in output & fieldname	Check this option if you want to have the name of the XML file to which the row belongs in the output stream. You can specify the name of the field where the filename will end up in.
Rownum in output & fieldname	Check this option if you want to have a row number (starts at 1) in the output stream. You can specify the name where the integer will end up in.
Include time in filename	Includes the system date in the filename (_235959).
Limit	You can specify the maximum number of rows to read here.
Nr of header rows to skip	Specify the number of rows to skip, from the start of an XML document, before starting to process.
Location	<p>Specify the path by way of elements to the repeating part of the XML file. For example if you are reading rows from this XML file:</p> <pre data-bbox="534 1639 1418 1798" style="background-color: #ffffcc; padding: 10px;"> &lt;Rows&gt; &lt;Row&gt;   &lt;Field1&gt;...&lt;/Field1&gt; ... &lt;/Row&gt; ... &lt;/Rows&gt; </pre> <p>Then you set the location to Rows, Row  <b>Note:</b> <i>you can also set the root (Rows) as a repeating element location. The output will then contain 1 (one) row.</i></p>

Option	Description
Fields	<ul style="list-style-type: none"> <li>✓ Name: Name of the field.</li> <li>✓ Type: Type of the field can be either String, Date or Number.</li> <li>✓ Format mask to convert with: See 8.5.1.5 for a complete description of format specifiers.</li> <li>✓ Length:               <ul style="list-style-type: none"> <li>○ Number: Total number of significant figures in a number;</li> <li>○ String: total length of string;</li> <li>○ Date: length of <i>printed</i> output of the string (e.g. 4 only gives back the year).</li> </ul> </li> <li>✓ Precision:               <ul style="list-style-type: none"> <li>○ Number: Number of floating point digits;</li> <li>○ String: unused;</li> <li>○ Date: unused.</li> </ul> </li> <li>✓ Currency: Symbol used to represent currencies like \$10,000.00 or €5.000,00</li> <li>✓ Decimal: A decimal point can be a "." (10,000.00) or "," (5.000,00)</li> <li>✓ Grouping: A grouping can be a "," (10,000.00) or "." (5.000,00)</li> <li>✓ Trim type: The trimming method to apply on the string found in the XML.</li> <li>✓ Repeat: Check this if you want to repeat empty values with the corresponding value from the previous row.</li> <li>✓ Position: The position of the XML element or attribute. You use the following syntax to specify the position of an element:  <u>The first element called "element":</u> E=element/1   <u>The first attribute called "attribute":</u> A=attribute/1   <u>The first attribute called "attribute" in the second "element" tag:</u>  E=element/2, A=attribute/1 </li> </ul> <p><b>NOTE:</b> You can auto-generate all the possible positions in the XML file supplied by using the "Get Fields" button.</p> <p><b>NOTE:</b> Support was added for XML documents where all the information is stored in the Repeating (or Root) element. The special R= locator was added to allow you to grab this information. The "Get fields" button finds this information if it's present.</p>

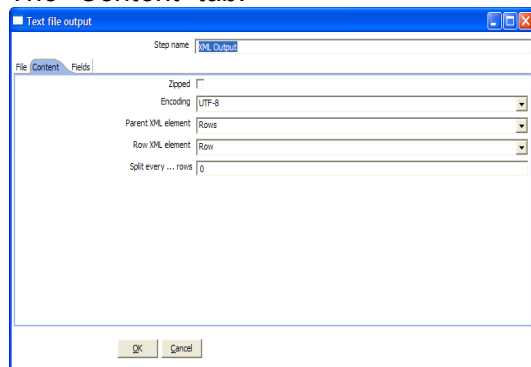
## 8.5.41. XML Output

### 8.5.41.1. Screenshot

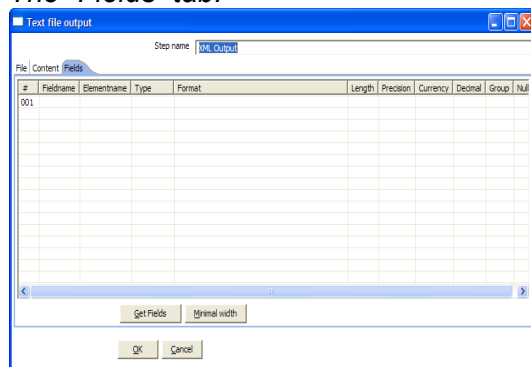
The “files” tab:



The “Content” tab:



The “Fields” tab:



### 8.5.41.2. Icon



### 8.5.41.3. General description

This step allows you to write rows from any source to one or more XML files.

### 8.5.41.4. Options

Option	Description
Step name	Name of the step. This name has to be unique in a single transformation.
Filename	This field specifies the filename and location of the output text file.
Extension	Adds a point and the extension to the end of the filename (.xml).
Include stepnr in filename	If you run the step in multiple copies (see also <a href="#">7.2. Launching several copies of a step</a> , the copy number is included in the filename, before the extension (_0).
Include date in filename	Includes the system date in the filename (_20041231).
Include time in filename	Includes the system date in the filename (_235959).
Split every ... rows	The maximum number of rows of data to put in a single XML file before another is created.
Zipped	Check this if you want the XML file to be stored in a ZIP archive.
Encoding	The encoding to use. This encoding is specified in the header of the XML file.
Parent XML element	The name of the root element in the XML document.
Row XML element	The name of the row element to use in the XML document.
Fields	<ul style="list-style-type: none"> <li>✓ Fieldname: Name of the field.</li> <li>✓ Elementname: the name of the element in the XML file to use.</li> <li>✓ Type: Type of the field can be either String, Date, or Number.</li> <li>✓ Format mask to convert with: see <a href="#">7.4.1.5.1. Number formats</a> for a complete description of format specifiers.</li> <li>✓ Length: Output string is padded to this length if it is specified.</li> <li>✓ Precision: The precision to use.</li> <li>✓ Currency: Symbol used to represent currencies like \$10,000.00 or €5.000,00</li> <li>✓ Decimal: A decimal point can be a "." (10,000.00) or "," (5.000,00)</li> <li>✓ Grouping: A grouping can be a "," (10,000.00) or "." (5.000,00)</li> <li>✓ Null: the string to use in case the field value is null.</li> </ul>

### 8.5.42. Delete

#### 8.5.42.1. Screenshot

[illegible]

#### 8.5.42.2. Icon

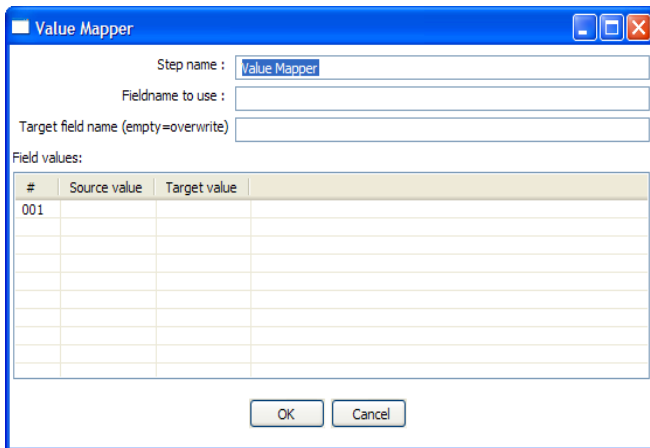


#### 8.5.42.3. General description

This step is the same as the [8.4.21. Update](#) step , except that instead of updating, rows are deleted.

## 8.5.43. Value Mapper

### 8.5.43.1. Screenshot



### 8.5.43.2. Icon



### 8.5.43.3. General description

This step simply maps string values from one value to another. Usually you will want to solve this problem by storing the conversion table in a database. However, this is the alternative: simply enter the conversion table into the Value Mapper dialog.

For example, if you want to replace language codes:

Fieldname to use: LanguageCode

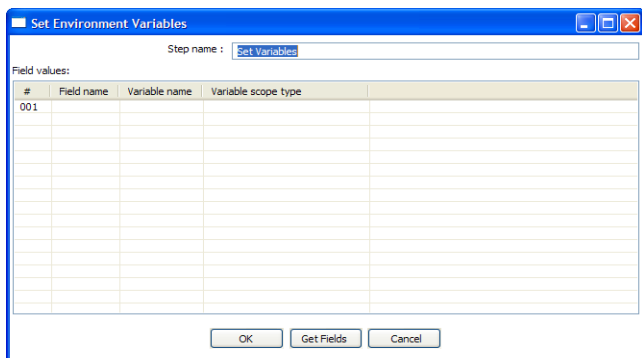
Target fieldname: LanguageDesc

Source/Target: EN/English, FR/French, NL/Dutch, ES/Spanish, DE/German, ...



## 8.5.44. Set Variable

### 8.5.44.1. Screenshot



### 8.5.44.2. Icon



### 8.5.44.3. General description

This step allows you to set variables in a job or in the virtual machine. It accepts one (and only one) row of data to set the value of a variable. Here are the possible scope settings:

- **Valid in the virtual machine:** the complete virtual machine will know about this variable.  
**WARNING:** this makes your transformation only fit to run in a stand-alone fashion. Running on an application server like on the Pentaho framework can become a problem. That is because other transformations running on the server will also see the changes this step makes.
- **Valid in the parent job:** the variable is only valid in the parent job.
- **Valid in the grand-parent job:** the variable is valid in the grand-parent job and all the child jobs and transformations.
- **Valid in the root job:** the variable is valid in the root job and all the child jobs and transformations.

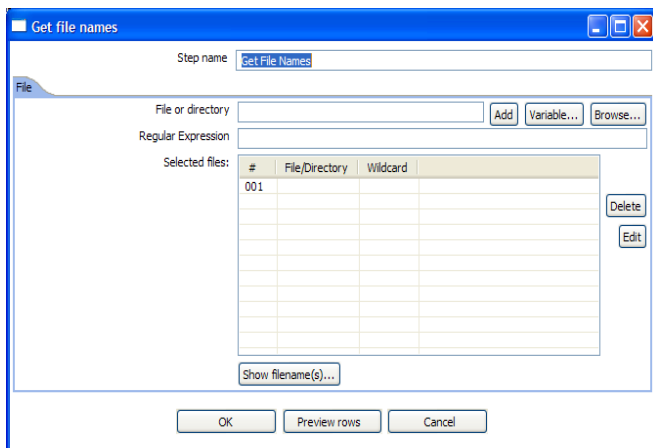
### 8.5.44.4. Variable usage

Refer to “**8.4 Use of variables**” on page 33 for a description of the use of variables.



## 8.5.46. Get Filename

### 8.5.46.1. Screenshot



### 8.5.46.2. Icon



### 8.5.46.3. General description

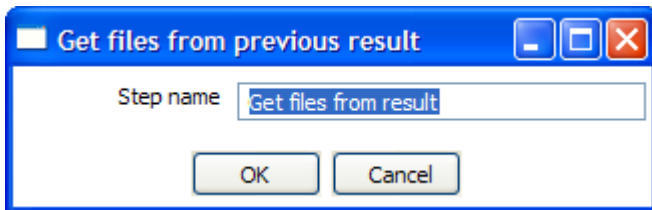
This step allows you to get information regarding filenames on the file system on your system. Please also see [8.4.1.4.1. Filename specification options](#) for a detailed description on how to use the specify filenames.

The output fields for this step are:

- **filename**: the complete filename, including the path (/tmp/kettle/somefile.txt)
- **short\_filename**: only the filename, without the path (somefile.txt)
- **path**: only the path (/tmp/kettle/)

## 8.5.47. Get files from result

### 8.5.47.1. Screenshot



### 8.5.47.2. Icon

Get  
Files

### 8.5.47.3. General description

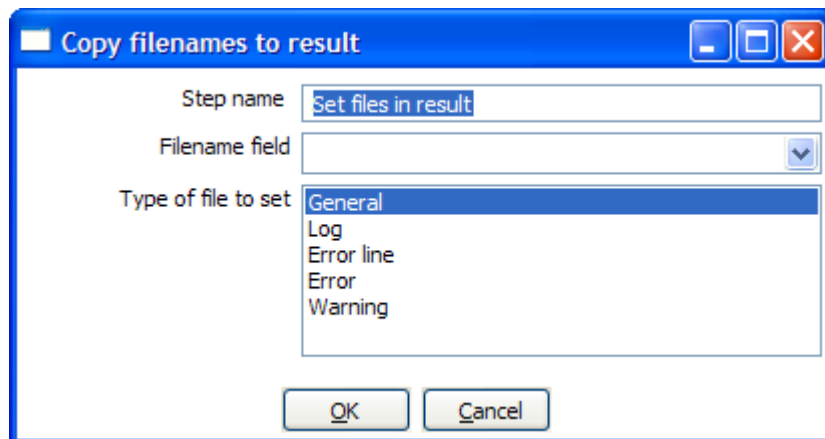
Every time a file gets processed, used or created in a transformation or a job, the details of the file, the job entry, the step, etc. is captured and added to the result. You can access this file information using this step.

These are the output fields:

Field name	Type	Example
type	String	Normal, Log, Error, Error-line, etc.
filename	String	somefile.txt
path	String	C:\Foo\Bar\somefile.txt
parentorigin	String	Process files transformation
origin	String	Text File Input
comment	String	Read by text file input
timestamp	Date	2006-06-23 12:34:56

## 8.5.48. Set files in result

### 8.5.48.1. Screenshot



### 8.5.48.2. Icon

Set  
Files

### 8.5.48.3. General description

In certain cases, we can use this step to steer the list of files in the result ourself.

For example the Mail job entry can use this list of files to attach to a mail, so perhaps you don't want all files sent, but only a certain selection. For this, you can create a transformation that sets exactly those files you want to attach.

## 8.5.49. Blocking step

### 8.5.49.1. Screenshot



### 8.5.49.2. Icon

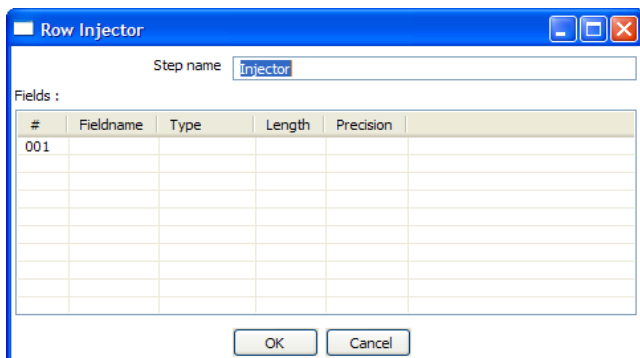


### 8.5.49.3. General description

This is again a very simple step. It blocks all output until the very last row was received from the previous step. This last row is then sent off to the next step. This step then knows that all previous steps have finished. You can use this for triggering custom plugin, stored procedures, java scripts, etc.

## 8.5.50. Injector

### 8.5.50.1. Screenshot



### 8.5.50.2. Icon



### 8.5.50.3. General description

- Injector was created for those people that are developing special purpose transformations and want to “Inject” rows into the transformation using the Kettle API and Java. Among other things you can build “headless” transformations with it: transformations that have no input at design time: don’t read from file or database.
- Here is some information on how to do it:

- You can ask a Trans object for a RowProducer object
- Also see the use case test in package: `be.ibridge.kettle.test.rowproducer`
- Use this type of code:

```
Trans trans = new Trans(... TransMeta ...);
trans.prepareExecution(args);
RowProcuder rp = trans.addRowProducer(String stepname, int stepCopy);
```

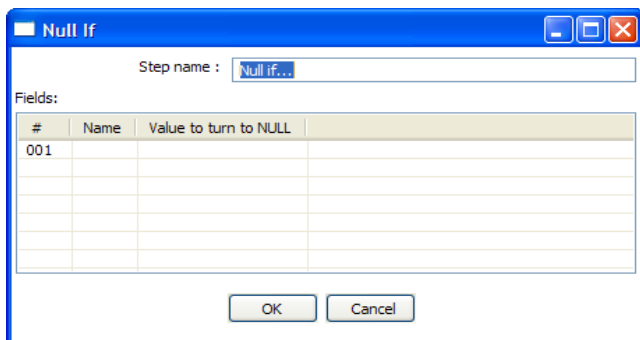
- After that you start the threads in the transformation. Then you can inject the rows while the transformation is running:

```
trans.startThreads();
...
rp.putRow(Row SomeRowYouHaveToInject);
...
```

- As you can see from the screenshot below, you can specify the rows you are expecting to be injected. This makes it easier to build transformations because you have the meta-data at design time.

## 8.5.51. Null If

### 8.5.51.1. Screenshot



### 8.5.51.2. Icon



### 8.5.51.3. General description

If the string representation of a certain field is equal to the specified value, then the value is set the null (empty).



## 9. GRAPHICAL VIEW

### 9.1. Description

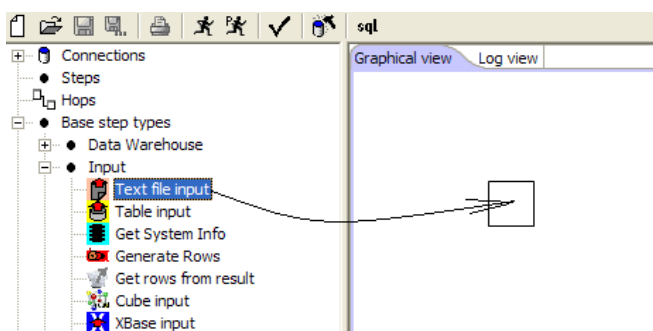
The Graphical View is the canvas on which transformations are drawn.

It shows an easy to understand representation of the work that needs to be done and the flow of the data.

### 9.2. Adding steps

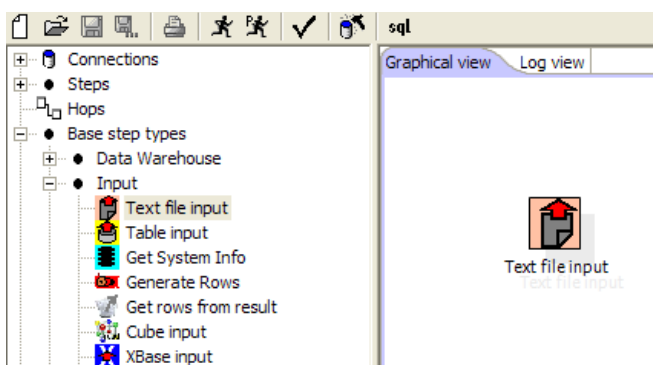
#### 9.2.1. Create steps by drag and drop

Adding steps to a transformation on the canvas is easy: simply select a step type from the tree on the left and drag in onto the canvas:



At the location of the mouse you will see a square that represents the location of the steps when you let go of the button.

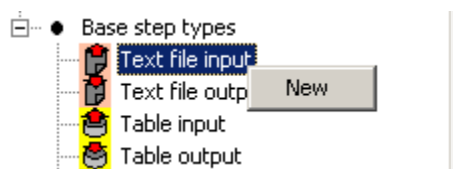
When you let go of the mouse button the selected step (Text file input) will become part of the transformation.



### 9.2.2. Create steps from the step types tree

Another way to create an extra step for your transformation is by selecting “New” step when you right click on the step type in the tree on the left.

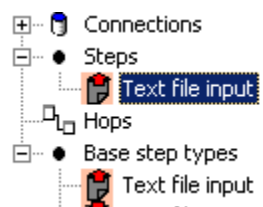
You can also just double-click on the step type.



After this, you will be shown a dialog where you can enter the settings. For the complete description of the dialogs and settings, please check out [7.4. Step Types](#).

When you press OK in the dialog a new step is created in the transformation.

You will see in this case however that the step is not drawn. That is of-course because we haven't told Spoon where to put it. We say that the step is not drawn but hidden.



If you drag this step onto the canvas, it will become drawn.

### 9.2.3. Create steps on the place where you like them

You can also just right click on the canvas and select “New step...”.

## 9.3. Hiding a step

If you right click on a step that is drawn on the graphical view, you will get a popup-menu that allows you to select the option: “Hide step”. This will remove the step from the graphical view, but not delete it.

## 9.4. Step options (popup menu)

### 9.5. Edit step

This opens the step dialog so that you can change its settings.

### 9.6. Edit description

This opens a dialog that allows you to enter a textual description of the step.

### 9.7. Copy data / distribute data

See [7.3. Distribute or copy?](#)

### 9.8. Copies

See [7.2. Launching several copies of a step](#)

### 9.9. Duplicate step

This option will create a copy, positioned a bit lower to the right of the original step.

### 9.10. Delete step

This will permanently remove the step from the transformation.

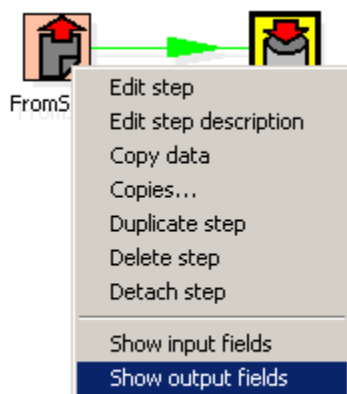
### 9.11. Show input fields

This option tries to determine all the fields and their origin by tracing the input-streams back to their source.

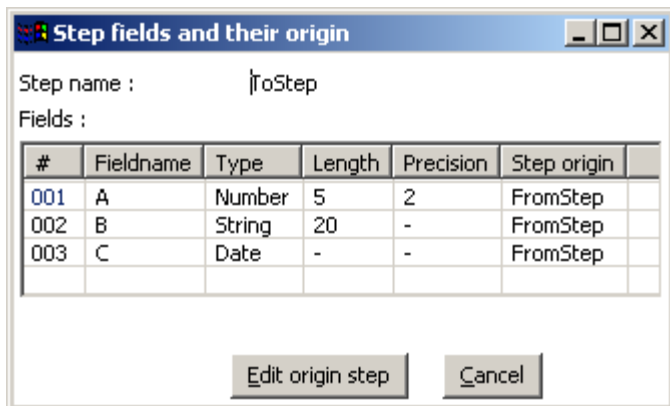
## 9.12. Show output fields

This option adds the fields of the current step to the ones of the input fields.

For example this might be an example:



The result could then possibly be:



## 9.13. Adding hops

On the graphical view the quickest way to create a new hop is by dragging with the mouse from one step to another using the middle button.

You can also drag the left button and press the control key at the same time.

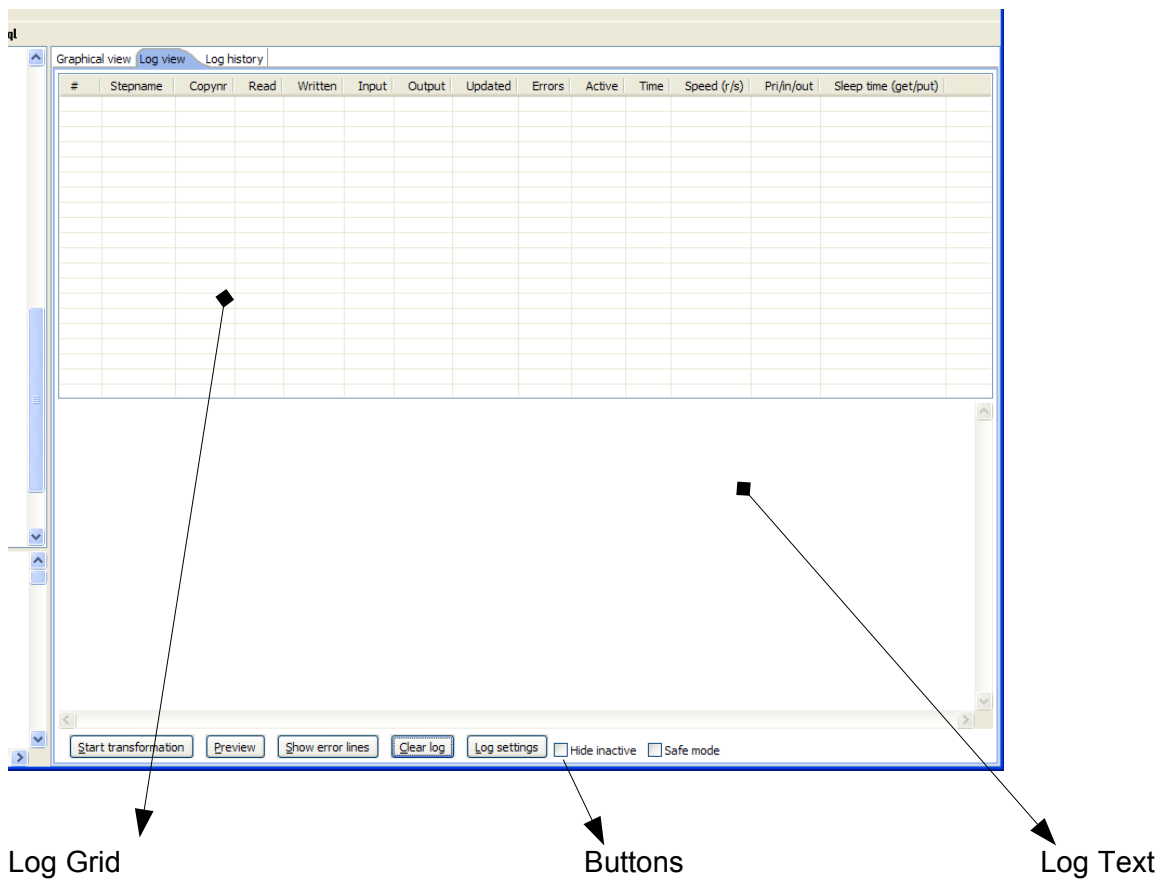
For a more complete explanation regarding hops, please check out [5. Hops](#)

## 10. LOG VIEW

### 10.1. Description

The Log View shows what's happening when a transformation is running. First of all it shows all the (copies of) the steps that are launched as a part of the transformation. Secondly, it shows the log as it would be shown if the transformation would be launched by Pan.

### 10.2. Screenshot



## 10.3. Log Grid

The log grid shows the following columns:

- ✓ The name of the step
- ✓ Copy number of the step
- ✓ Number of lines read from input-streams
- ✓ Number of lines written to output-streams
- ✓ Number of lines read from file or database
- ✓ Number of lines written to file or database
- ✓ Number of lines updated in the database
- ✓ Number of errors that occurred
- ✓ The status of the step: running, finished or stopped
- ✓ The number of seconds that the step has been running.
- ✓ The speed in rows per second at which the step processes rows.
- ✓ Priority of the step (10=highest, 1=lowest), nr of rows in the input-stream(s), nr of rows in the output-stream(s).
- ✓ Sleep time (get/put) is the time that a step had to go to sleep (in nano seconds) because the input buffer was empty (get) or the output buffer was full (put).

**NOTE:** *The system is tuning the steps priority in such a way that the slowest steps get the highest priority.*

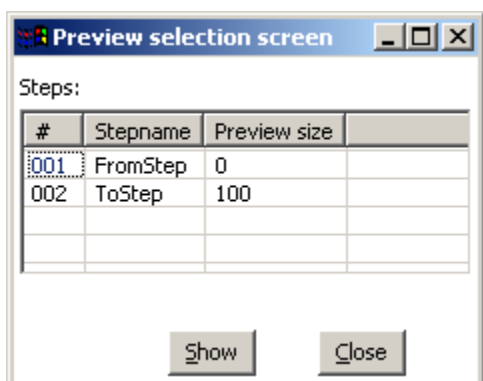
## 10.4. Buttons

### 10.4.1. Start transformation

This button starts the transformation you're designing. Please note that Spoon tries to launch this as Pan would: from XML-file or repository. It is therefore necessary that the transformation is saved. The output of the execution is displayed in the Log Text part of the Log View.

### 10.4.2. Preview

This buttons starts the transformation as it is defined in memory at the time you press the Preview button. The output of the execution is displayed in the Log Text part of the Log View. After you press the button you will be presented with a dialog like this:



Here you can enter the rows per step you want to preview. The preview size is automatically set to 100 rows for the steps that are selected. (Note that you can change this, see also: [2.14.1.2. Default preview size](#)) If no step is selected, the previous setting is taken.

After the requested rows are obtained from the different steps, the transformation is ended. After the transformation has ended, the result is shown.

### 10.4.3. Show error lines

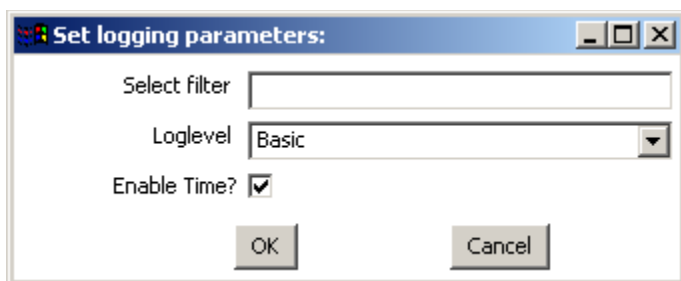
This option will show you all lines of the Log Text that contain the word ERROR (lower- or uppercase). You can then choose to edit the source step of the error.

### 10.4.4. Clear log

This clears the text in the Log Text Window.

### 10.4.5. Log Settings

This is the “Log Settings” dialog:



If you put a text in the filter field, only the lines that contain this text will be shown in the Log Text window.

The “Log level” setting allows you to select the logging level. You can choose one of these:

- ✓ Nothing: Don't show any output
- ✓ Error: Only show errors
- ✓ Minimal: Only use minimal logging
- ✓ Basic: This is the default basic logging level
- ✓ Detailed: Give detailed logging output
- ✓ Debug: For debugging purposes, very detailed output.
- ✓ Row level: Logging at a row level, this can generate a lot of data.

If the “Enable time” option is enabled, all lines in the logging will be preceded by the time of day.

#### **10.4.6. Hide inactive**

Checking this hides all steps that finished processing.

#### **10.4.7. Safe mode**

Places the transformation in Safe Mode. Additional row checking is enabled at runtime, see also: [2.20. Safe mode](#)



## 11. GRIDS

### 11.1. Description

Grids are used everywhere in Spoon & Kettle. They are used to enter or display information.

### 11.2. Functions

Most of the functions available in a grid (that is not read-only) are available by right clicking with the mouse on a grid:

- ✓ Insert before this row: Inserts an empty row before the row you clicked on.
- ✓ Insert after this row: Inserts an empty row after the row you clicked on.
- ✓ Move the row up: Move the row you clicked on up. The keyboard shortcut for this is CTRL-UP
- ✓ Move the row down: Move the row you clicked on down. The keyboard shortcut for this is CTRL-DOWN.
- ✓ Optimal column size including header: Resize all columns so that it displays all values completely, including the header. The keyboard shortcut for this function is function key F3.
- ✓ Optimal column size excluding header: Resize all columns so that it displays all values completely. The keyboard shortcut for this function is function key F4.
- ✓ Clear all: Clears all information in the grid. You will be asked to confirm this operation.
- ✓ Select all rows: Selects all rows in the grid. The keyboard shortcut for this function is CTRL-A.
- ✓ Clear selection: Clears the selection of rows in the grid. The keyboard shortcut for this function is ESC.
- ✓ Copy selected lines to clipboard: Copies the selected lines to the clipboard in a textual representation. These lines can then be exchanged with other programs such as spreadsheets or even other Spoon & Kettle dialogs. The keyboard shortcut for this function is CTRL-C.
- ✓ Paste clipboard to table: Insert the lines that are on the clipboard to the grid, right after the line on which you clicked. The keyboard shortcut for this function is CTRL-V.
- ✓ Cut selected lines: Copies the selected lines to the clipboard in a textual representation. After that, the lines are deleted from the grid. The keyboard shortcut for this function is CTRL-X.
- ✓ Delete selected lines: Deletes all selected lines from the grid. The keyboard shortcut for this function is DEL.
- ✓ Keep only selected lines: If there are more lines to delete than there are to keep, simply select the lines you want to keep and use this function. The keyboard shortcut for this function is CTRL-K.
- ✓ Copy field values to all rows: If all rows in the grid need to have the same value for a certain column, you can use this function to do this.
- ✓ Undo: Undo the previous grid operation. The keyboard shortcut for this function is CTRL-Z.
- ✓ Redo: Redo the next grid operation. The keyboard shortcut for this function is CTRL-Y.

### 11.3. Navigating

If you click on a cell in a grid, you can edit this field.

After pressing enter, you can navigate the grid by using the cursor keys.

Pressing enter again allows you to edit a field.

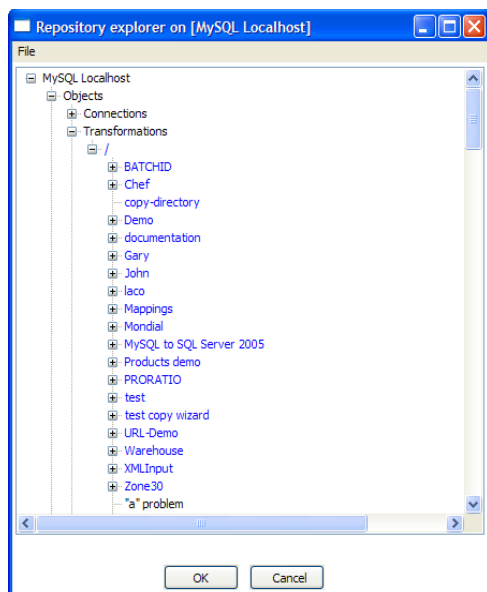
## 12. REPOSITORY EXPLORER

---

### 12.1. Description

The repository Explorer shows you a tree view on the database repository to which you are connected. It allows you to examine and modify the content.

### 12.2. Screenshot

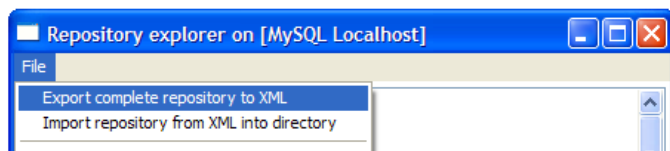


### 12.3. Functions

Check out the different functions when right clicking in the tree. It's fairly straightforward.

## 12.4. Backup / Recovery

It is possible to export the complete repository in XML: See the options available in the file menu of the repository explorer.



**NOTE:** *you can restore the objects from a backed up repository anywhere in the target repository directory tree.*

## 13. APENDIX A

### GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.  
 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA  
 Everyone is permitted to copy and distribute verbatim copies  
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts  
 as the successor of the GNU Library Public License, version 2, hence  
 the version number 2.1.]

#### Preamble

The licenses for most software are designed to take away your  
 freedom to share and change it. By contrast, the GNU General Public  
 Licenses are intended to guarantee your freedom to share and change  
 free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some  
 specially designated software packages--typically libraries--of the  
 Free Software Foundation and other authors who decide to use it. You  
 can use it too, but we suggest you first think carefully about whether  
 this license or the ordinary General Public License is the better  
 strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use,  
 not price. Our General Public Licenses are designed to make sure that  
 you have the freedom to distribute copies of free software (and charge  
 for this service if you wish); that you receive source code or can get  
 it if you want it; that you can change the software and use pieces of  
 it in new free programs; and that you are informed that you can do  
 these things.

To protect your rights, we need to make restrictions that forbid  
 distributors to deny you these rights or to ask you to surrender these  
 rights. These restrictions translate to certain responsibilities for  
 you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis  
 or for a fee, you must give the recipients all the rights that we gave  
 you. You must make sure that they, too, receive or can get the source  
 code. If you link other code with the library, you must provide  
 complete object files to the recipients, so that they can relink them  
 with the library after making changes to the library and recompiling  
 it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the  
 library, and (2) we offer you this license, which gives you legal  
 permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that  
 there is no warranty for the free library. Also, if the library is  
 modified by someone else and passed on, the recipients should know  
 that what they have is not the original version, so that the original  
 author's reputation will not be affected by problems that might be  
 introduced by others.

Finally, software patents pose a constant threat to the existence of  
 any free program. We wish to make sure that a company cannot  
 effectively restrict the users of a free program by obtaining a  
 restrictive license from a patent holder. Therefore, we insist that  
 any patent license obtained for a version of the library must be  
 consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the  
 ordinary GNU General Public License. This license, the GNU Lesser  
 General Public License, applies to certain designated libraries, and  
 is quite different from the ordinary General Public License. We use  
 this license for certain libraries in order to permit linking those  
 libraries into non-free programs.

When a program is linked with a library, whether statically or using

a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

#### GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that

you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.


In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany

User manual Last updated: 12-07-2006	Pentaho Data Integration	 pentaho™ open source business intelligence™
	Spoon 2.3.0	

it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.


If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials

User manual Last updated: 12-07-2006	Pentaho Data Integration	 pentaho™ open source business intelligence™
	Spoon 2.3.0	

specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.


8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.



User manual Last updated: 12-07-2006	Pentaho Data Integration	 pentaho™ open source business intelligence™
	Spoon 2.3.0	

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY


15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### END OF TERMS AND CONDITIONS

#### How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest

User manual Last updated: 12-07-2006	Pentaho Data Integration	 pentaho™ open source business intelligence™
	Spoon 2.3.0	

possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.
```

```
This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.
```

```
You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library `Frob' (a library for tweaking knobs) written by James Random Hacker.
```

```
<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!