Pentaho Data Integration

Spoon 2.5.0 User Guide

Last Modified on May 13th, 2007

1. Contents

2. About This Document

2.1. What it is

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

2.2. What it is not

In this document we do not describe in great detail on how to create jobs and 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.

2.3. Other documentation

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

Flash demos, screen shots, and an introduction to building a simple transformation:
http://kettle.pentaho.org/screenshots/

Pentaho Data Integration community website – news, case studies, weekly tips and more:
http://kettle.pentaho.org

Pentaho Data Integration Forum – discussions on design, features, bugs and enhancements:
http://forums.pentaho.org/forumdisplay.php?f=69

Running transformations in batch using Pan:	Pan-2.5.pdf
Running jobs in batch using Kitchen:	Kitchen-2.5.pdf

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)

3. Introduction to Spoon

3.1. 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 ETTL needs: the Extraction, Transformation, Transportation and Loading of data.

Spoon is a graphical user interface that allows you to design transformations and jobs that can be run with the Kettle tools Pan (transformations) and Kitchen(jobs). 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. Kitchen is a program that can execute jobs designed by Chef in XML or in a database repository. Usually jobs are scheduled in batch mode to be run automatically at regular intervals.

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

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

In short: **Pentaho Data Integration makes data warehouses easier to build, update and maintain!**

3.2. 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-2.5.0.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
```

3.3. 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.

3.4. 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

3.5. Known issues

Linux

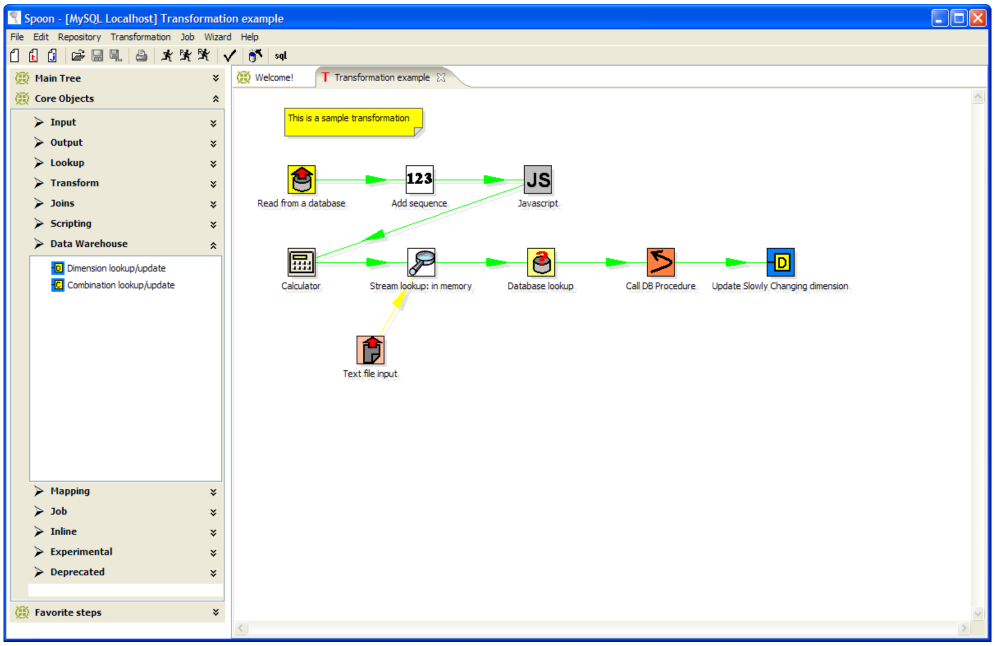
Occasional JVM crashes running SuSE Linux and KDE. Running under Gnome has no problems. (detected on SUSE Linux 10.1 but earlier versions suffer the same problem)

FreeBSD

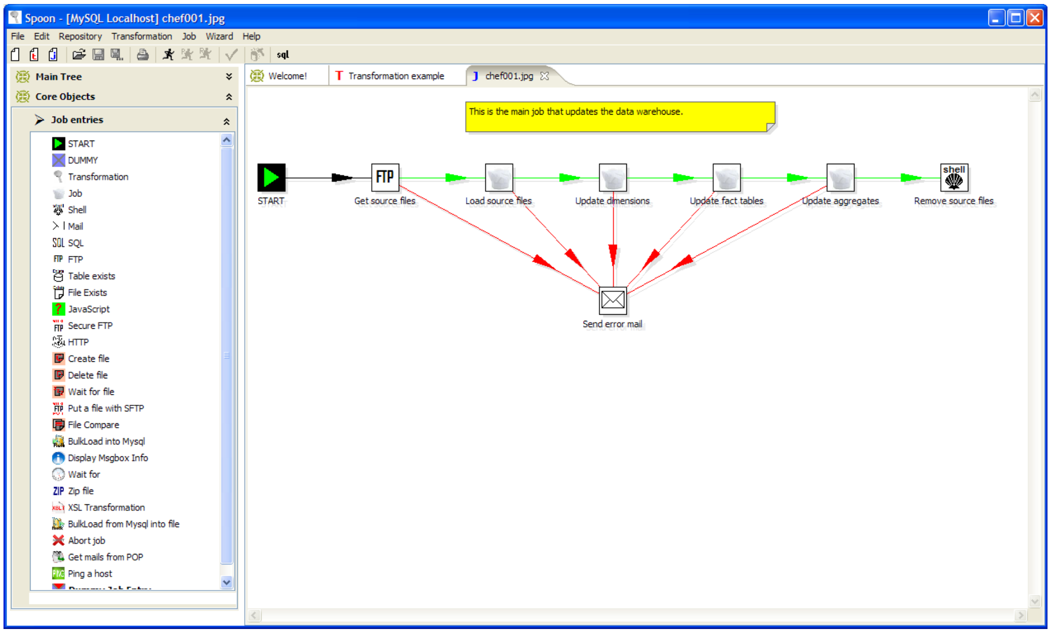
Problems with drag and drop. Workaround is to use the right click popup menu on the canvas. (Insert new step)

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

3.6. Screen shots



Designing a Transformation



Designing a job

The upper left panel of Spoon allows you to browse connections along with the jobs and transformations you currently have open. When designing a transformation, the lower left panel contains the available steps used to build your transformation including input, output, lookup, transform, joins, scripting steps and more. When designing a job, the component tree contains the available job entries. When designing a job, the

Core Objects bar contains core job entry types. These items are described in detail in the chapters below: 4. Database Connections, 7. Hops, 10. Transformation Steps, 12. Job Entries, 13. Graphical View.

3.7. Command line options

These are the command line options that you can use.

Important Notes:

- *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.*
- *It's important that if spaces are present in the option values, you use quotes or double quotes to keep them together. Take a look at the examples below for more info.*

These are the command line options that you can 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
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

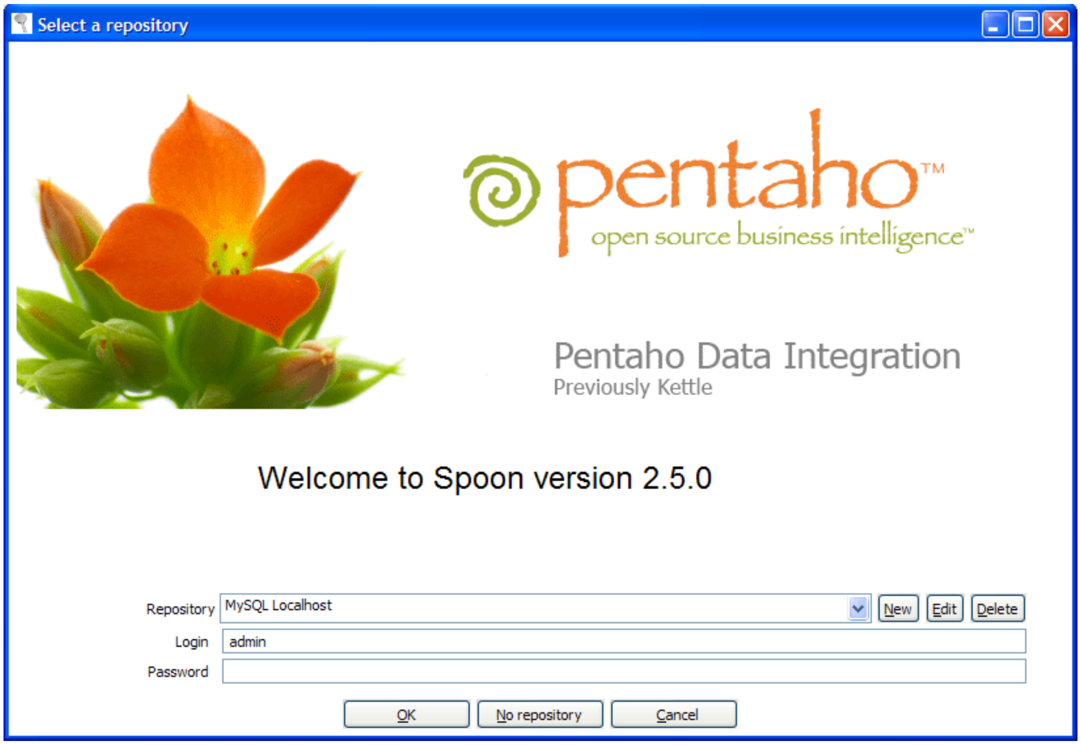
```
-job=Job Name
```

Use this option to select the job 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.*

3.8. 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.



Welcome to Spoon version 2.5.0

Repository: MySQL Localhost
Login: admin
Password:

The Repository login screen

The information concerning repositories is stored in a 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 2.14. 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 or the “Repository/Edit User” menu.*

3.9. 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.***

3.10. 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.*

3.11. 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".

3.12. Definitions

3.12.1. Transformation 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.

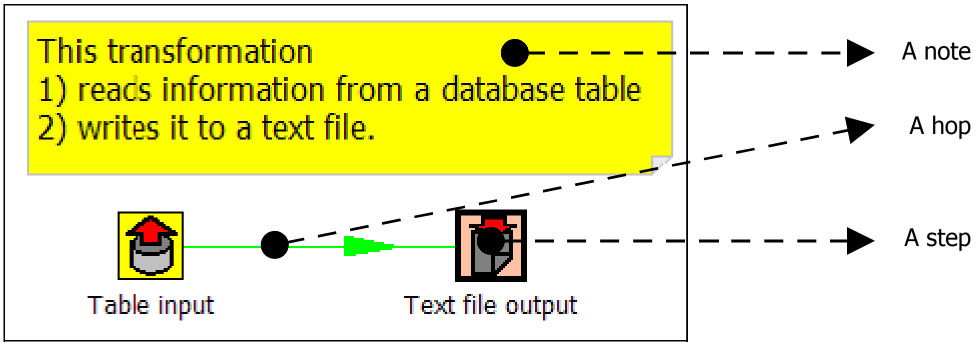
Row: a row exists of 0 or more values

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

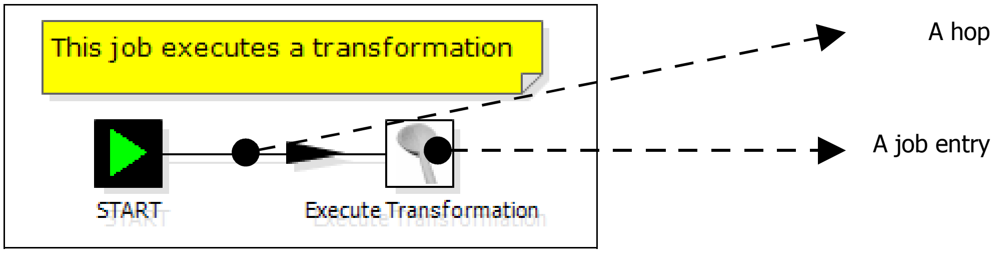

This transformation
1) reads information from a database table
2) writes it to a text file.
Table input
Text file output
A note
A hop
A step

3.12.2. Job Definitions

Job Entry: A job entry is one part of a job and performs a certain

Hop: a hop is a graphical representation of one or more data streams between 2 steps. A hop always represents the link between two job entries and can be set (depending on the type of originating job entry) to execute the next job entry unconditionally, after successful execution or failed execution.

Note: *a note is a piece of information that can be added to a job*



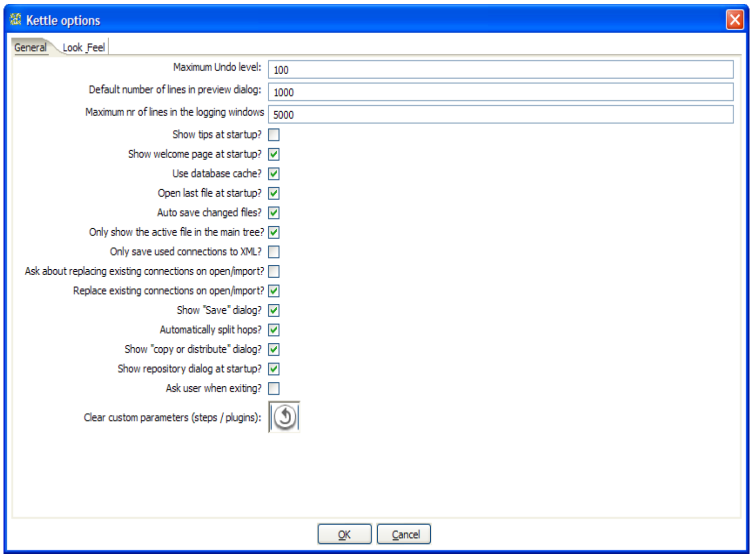
3.13. Toolbar

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

	Meaning
🗋	Create a new transformation or job
t	Create a new transformation
j	Create a new job
📂	Open transformation/job from file if you're not connected to a repository or from the repository if you are connected to one.
💾	Save the transformation/job to a file or to the repository.
💾	Save the transformation/job 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/job: runs the current transformation from XML file or repository.
P🏃	Preview transformation: runs the current transformation from memory. You can preview the rows that are produced by selected steps.
R🏃	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.
sql	Generate the SQL that is needed to run the loaded transformation.

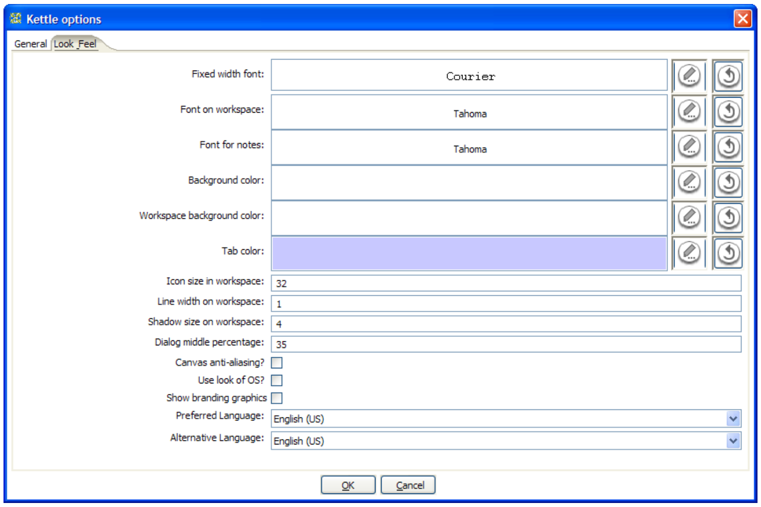
3.14. Options

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:



Maximum Undo level: 100
Default number of lines in preview dialog: 1000
Maximum nr of lines in the logging windows 5000
Show tips at startup? □
Show welcome page at startup? ☑
Use database cache? ☑
Open last file at startup? ☑
Auto save changed files? ☑
Only show the active file in the main tree? ☑
Only save used connections to XML? □
Ask about replacing existing connections on open/import? □
Replace existing connections on open/import? ☑
Show "Save" dialog? ☑
Automatically split hops? ☑
Show "copy or distribute" dialog? ☑
Show repository dialog at startup? ☑
Ask user when exiting? □
Clear custom parameters (steps / plugins):

The "general" options tab



Fixed width font: Courier
Font on workspace: Tahoma
Font for notes: Tahoma
Background color:
Workspace background color:
Tab color:
Icon size in workspace: 32
Line width on workspace: 1
Shadow size on workspace: 4
Dialog middle percentage: 35
Canvas anti-aliasing? □
Use look of OS? □
Show branding graphics □
Preferred Language: English (US)
Alternative Language: English (US)

The "look and feel" options tab

3.14.1. General tab

3.14.1.1. Maximum Undo Level

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

3.14.1.2. Default preview size

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

3.14.1.3. Show tips at startup

This option sets the display of tips at startup.

3.14.1.4. Show welcome page at startup?

This option will control whether or not to display the welcome page when launching spoon.

3.14.1.5. 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 3rd party tools, clearing the database cache manually may be necessary.*

3.14.1.6. 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.

3.14.1.7. Auto save changed file

This option automatically saves a changed transformation before running.

3.14.1.8. Only show the active file in the main tree

This option reduces the number of transformation and job items in the main tree on the left by only showing the currently active file.

3.14.1.9. 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.

3.14.1.10. Ask about replacing existing connections on open/import

This option asks before replacing existing database connections during import.

3.14.1.11. Replace existing connections on open/import

This is the action that's being taken when there is no dialog shown. (see previous option)

3.14.1.12. Show save confirmation

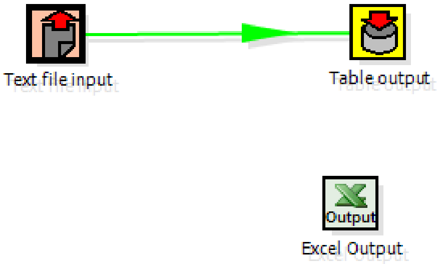
This flag allows you to turn off the confirmation dialogs you receive when a transformation has been changed.

3.14.1.13. Automatically split hops

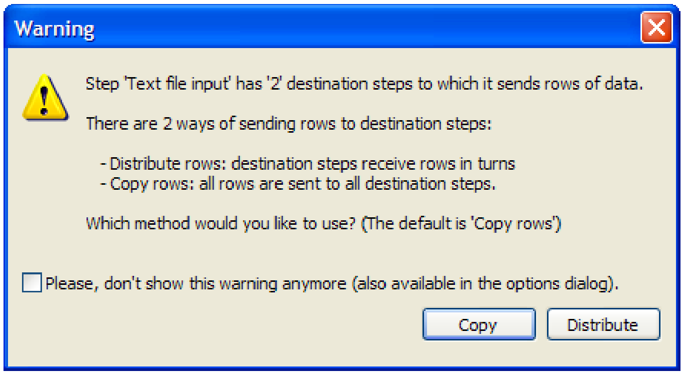
This option turns off the confirmation dialogs you get when you want to split a hop. (see also 7.4. Splitting A Hop)

3.14.1.14. Show copy or distribute rows warning?

This option turns off the warning message that appears when you link a step to multiple outputs. For example, if you have a transformation like this:



At the time you link step "Test file input" to "Excel Output", you will receive the following warning:


Warning

Step 'Text file input' has '2' destination steps to which it sends rows of data.

There are 2 ways of sending rows to destination steps:

- Distribute rows: destination steps receive rows in turns
- Copy rows: all rows are sent to all destination steps.

Which method would you like to use? (The default is 'Copy rows')

□ Please, don't show this warning anymore (also available in the options dialog).

Copy or distribute warning

3.14.1.15. show the repositories dialog at startup

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

3.14.1.16. Clear custom flags and parameters

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

3.14.2. Look & Feel tab

3.14.2.1. Default font

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

3.14.2.2. Graph font

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

3.14.2.3. Grid font

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

3.14.2.4. Note font

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

3.14.2.5. Background color

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

3.14.2.6. Graph background color

Sets the background color in the Graphical View of Spoon.

3.14.2.7. Color of selected tabs

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

3.14.2.8. 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.

3.14.2.9. Line width

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

3.14.2.10. Shadow size

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

3.14.2.11. 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.

3.14.2.12. 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. If you enable this and your environment doesn't work any more afterwards, change the value for option "EnableAntiAliasing" to "N" in file $HOME/.kettle/.spoonrc (C:\Documents and Settings\<user>\.kettle\.spoonrc on Windows)

3.14.2.13. 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.

3.14.2.14. Show branding graphics?

Enabling this option will draw Pentaho Data Integration branding graphics on the canvas and in the left hand side "expand bar".

3.14.2.15. Default locale to use

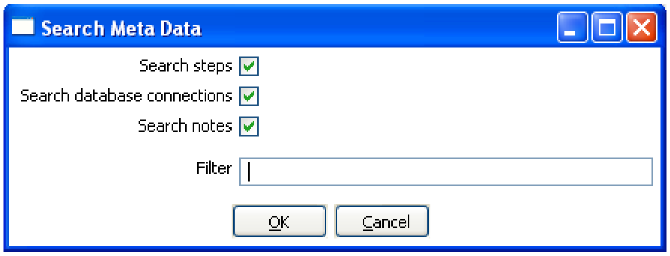
Here you can specify the default language setting. If a certain text hasn't been translated into this locale, Kettle will fall back to the fail over locale.

3.14.2.16. 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.

3.15. Search meta-data

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



Search steps ☑

Search database connections ☑

Search notes ☑

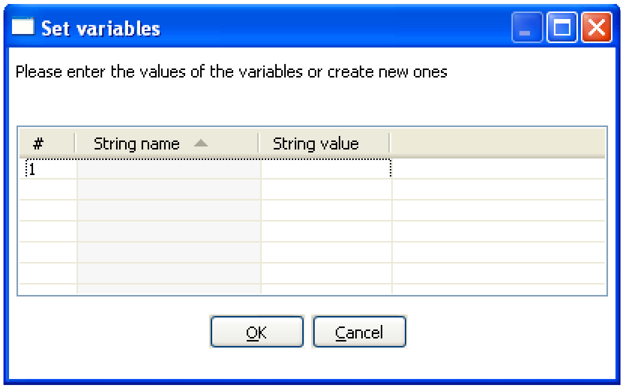
Filter

Search Meta Data Dialog

This option will search in any available fields, connectors or notes of all loaded jobs and transformations.   A detailed result is shown.

3.16. 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.



Please enter the values of the variables or create new ones

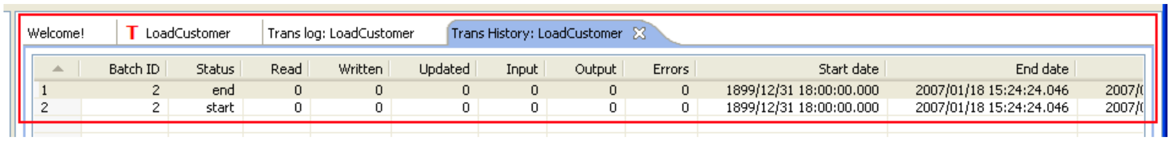
#	String name	String value
1		

Set variables dialog

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.

3.17. Execution log history

If you store the logging of the transformation or job in a database table (using the Transformations| Settings: logging tab or Job|Settings) you can see the overview of the runs by opening the history view. To open the history for a transformation, right-clicking on the transformation in the tree view to the left and select 'Open History View':



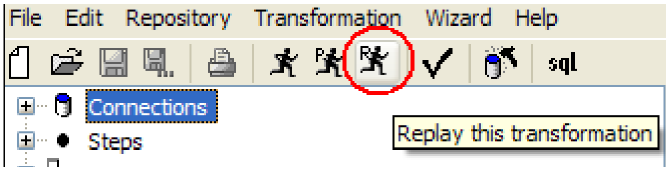
	Batch ID	Status	Read	Written	Updated	Input	Output	Errors	Start date	End date	
1	2	end	0	0	0	0	0	0	1899/12/31 18:00:00.000	2007/01/18 15:24:24.046	2007/(
2	2	start	0	0	0	0	0	0	1899/12/31 18:00:00.000	2007/01/18 15:24:24.046	2007/(

Transformation History Tab

NOTE: The log history for a job or transformation will also open by default each next time you open or execute the file.

3.18. Replay

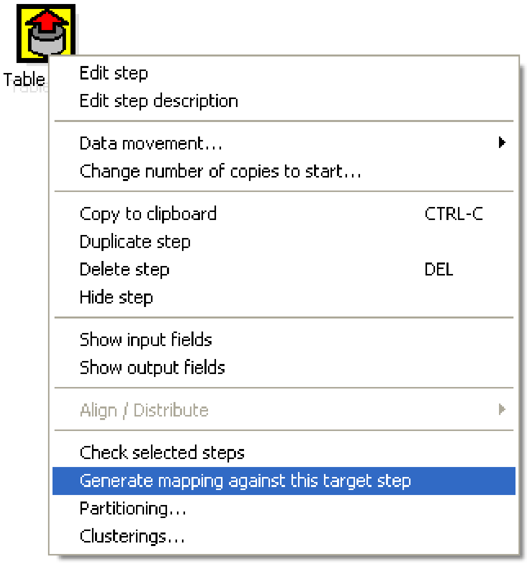
The Replay feature allows you to re-run a transformation that failed. Replay functionality is implemented for Text File Input and Excel input. It allows you to send files that had errors back to the source and have the data corrected. ONLY the lines that failed before are then processed during the replay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



Replay transformation

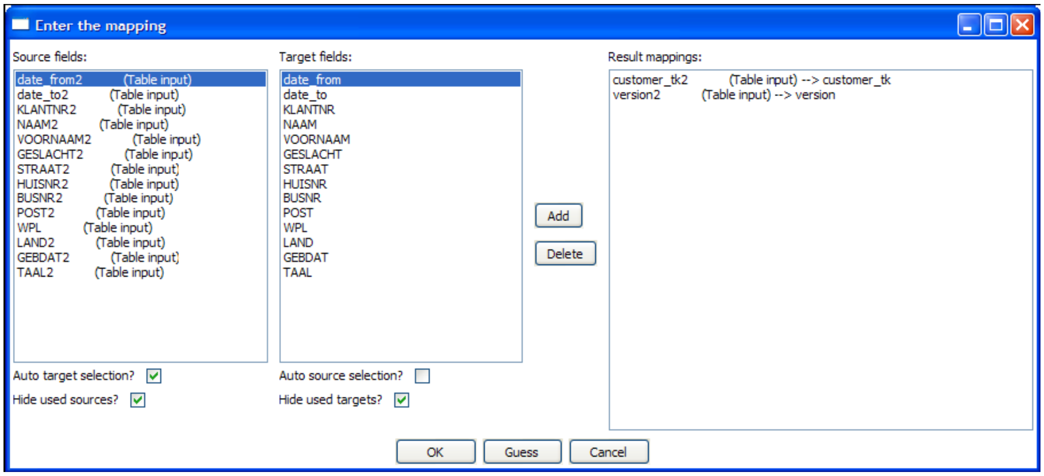
3.19. Generate mapping against target step

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



Generate Mapping Against Target Step

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



Source fields:

```
date_from2        (Table input)
date_to2          (Table input)
KLANTNR2          (Table input)
NAAM2             (Table input)
VOORNAAM2         (Table input)
GESLACHT2         (Table input)
STRAAT2           (Table input)
HUISNR2           (Table input)
BUSNR2            (Table input)
POST2             (Table input)
WPL               (Table input)
LAND2             (Table input)
GEBDAT2           (Table input)
TAAL2             (Table input)
```

Auto target selection? ☑

Hide used sources? ☑

Target fields:

```
date_from
date_to
KLANTNR
NAAM
VOORNAAM
GESLACHT
STRAAT
HUISNR
BUSNR
POST
WPL
LAND
GEBDAT
TAAL
```

Auto source selection? ☐

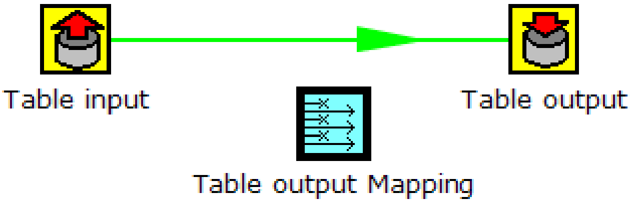
Hide used targets? ☑

Result mappings:

```
customer_tk2      (Table input) --> customer_tk
version2          (Table input) --> version
```

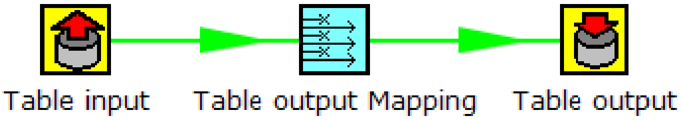
Generate Mapping Dialog

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:



Split hop before

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



Split hop after

3.20. Safe mode

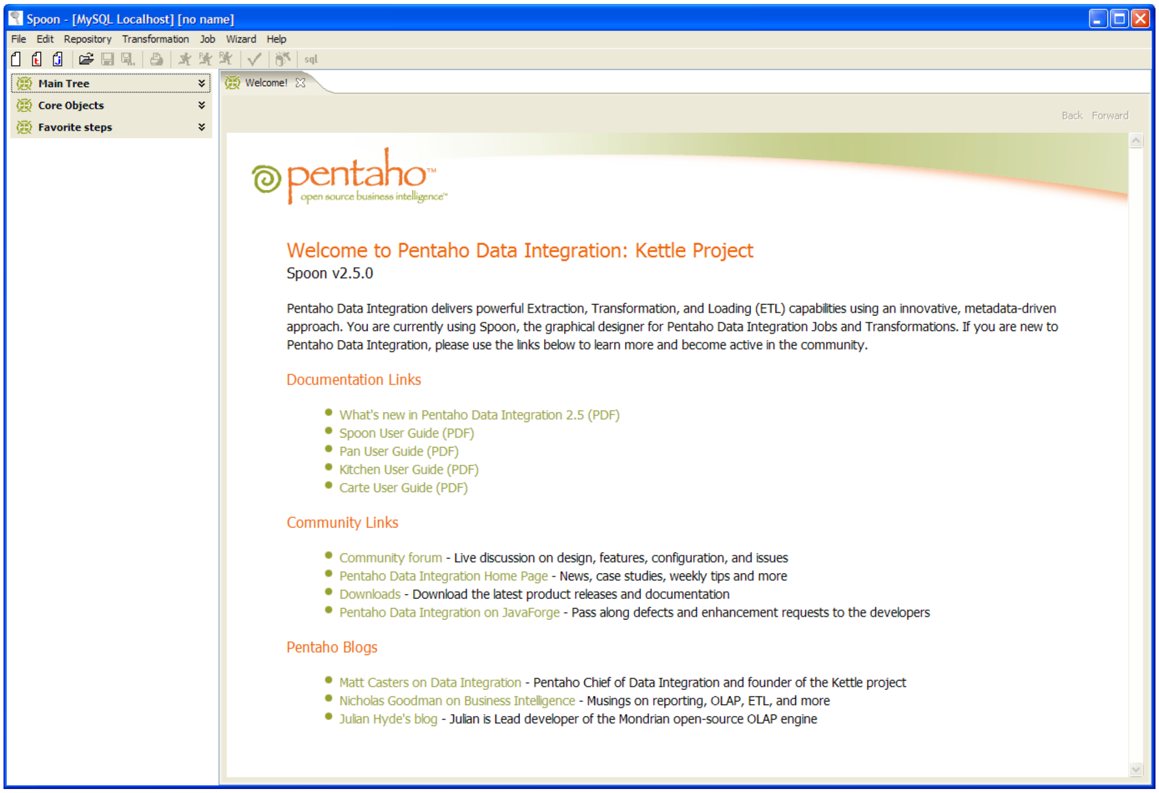
In cases where you are mixing the rows from various sources, you need to make sure that these row all have the same layout in all conditions. For this purpose, we added a "safe mode" option that is available in the Spoon logging window or on the Execute a Transformation/Job window. When running in "safe 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 "safe mode" option.*

3.21. Welcome Screen

The welcome screen will display the first time you launch Spoon 2.5.0 providing you with links to additional information about Pentaho Data Integration. You can disable the launching of the welcome page in using options (Edit|Options).


Spoon - [MySQL Localhost] [no name]
File Edit Repository Transformation Job Wizard Help
Main Tree
Core Objects
Favorite steps
Welcome!
Back Forward
pentaho
open source business intelligence

Welcome to Pentaho Data Integration: Kettle Project

Spoon v2.5.0

Pentaho Data Integration delivers powerful Extraction, Transformation, and Loading (ETL) capabilities using an innovative, metadata-driven approach. You are currently using Spoon, the graphical designer for Pentaho Data Integration Jobs and Transformations. If you are new to Pentaho Data Integration, please use the links below to learn more and become active in the community.

Documentation Links

- What's new in Pentaho Data Integration 2.5 (PDF)
- Spoon User Guide (PDF)
- Pan User Guide (PDF)
- Kitchen User Guide (PDF)
- Carte User Guide (PDF)

Community Links

- Community forum - Live discussion on design, features, configuration, and issues
- Pentaho Data Integration Home Page - News, case studies, weekly tips and more
- Downloads - Download the latest product releases and documentation
- Pentaho Data Integration on JavaForge - Pass along defects and enhancement requests to the developers

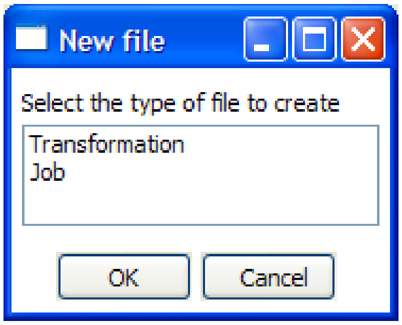
Pentaho Blogs

- Matt Casters on Data Integration - Pentaho Chief of Data Integration and founder of the Kettle project
- Nicholas Goodman on Business Intelligence - Musings on reporting, OLAP, ETL, and more
- Julian Hyde's blog - Julian is Lead developer of the Mondrian open-source OLAP engine

The welcome screen

4. Creating a Transformation or Job

To create a new Transformation or Job, either click the New button on the Toolbar, Select “File|New” from the menu:



Select the type of file to create

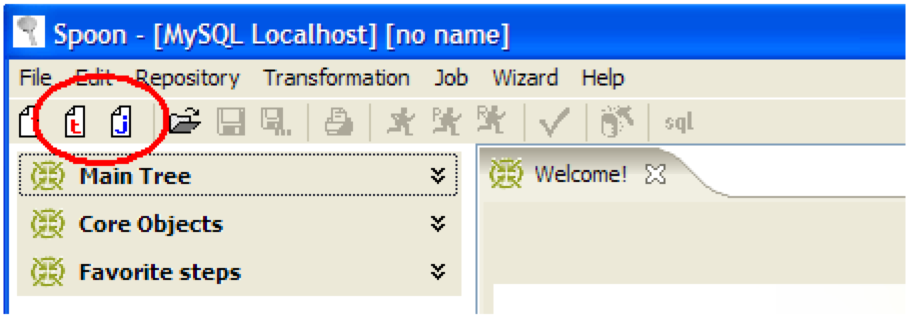
Transformation
Job

Selecting the type of file to create

After selecting the type of file you wish to create, a new tab will open showing the graphical view of your job or transformation (See 13. Graphical View for more details).

Icon	Description
J	Graphical View Tab for a Job
T	Graphical View Tab for a Transformation

Shortcuts have been made in the toolbar to create new transformations and jobs easier and faster:



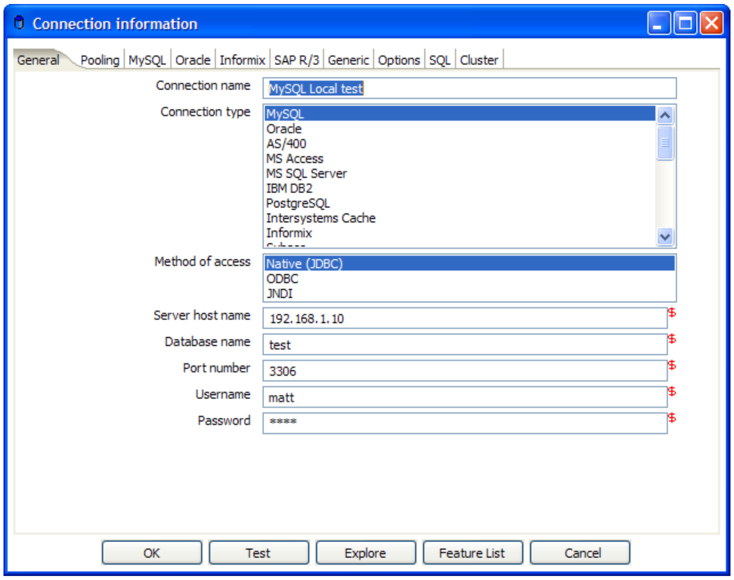
Create new files using the toolbar

5. Database Connections

5.1. 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.

5.2. Screen shot



Connection name: MySQL Local test

Connection type: MySQL
Oracle
AS/400
MS Access
MS SQL Server
IBM DB2
PostgreSQL
Intersystems Cache
Informix

Method of access: Native (JDBC)
ODBC
JNDI

Server host name: 192.168.1.10

Database name: test

Port number: 3306

Username: matt

Password: ****

The database connection dialog

5.3. Options

Connection name: a connection name uniquely name defines a connection across transformations and jobs.

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.

5.4. Quoting

We had more and more people complain about the handling of reserved words, field names with spaces in it, field names with decimals (.) in it, table names with dashes and other special characters in it ... we implemented a database specific quoting system that allows you to pretty much use any name or character that the database is comfortable with.

Pentaho Data Integration contains a list of reserved words for many (but not all) of the supported databases.

To correctly implement quoting, we had to go for a strict separation between the schema (user/owner) of a table and the tablename itself. Otherwise it would be impossible to properly quote tables or fields with one or more periods in them. Putting dots in table and field names is apparently common practice in certain ERP systems. (for example fields like "V.A.T.")

Because we too can be wrong when doing the quoting, we have added a new rule in version 2.5.0: when there is a start or end-quote in the tablename or schema, Pentaho Data Integration refrains from doing the quoting. This allows you to specify the quoting mechanism yourself. This leaves you all the freedom you need to get out of any sticky situation that might be left. Nevertheless, feel free to let us know about it so that we can improve our quoting algorithms.

5.5. Database Usage Grid

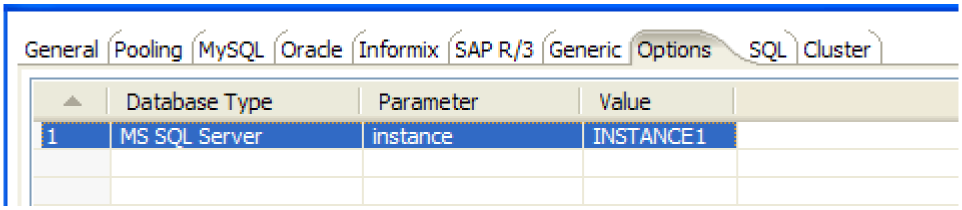
Database	Access Method	Server Name or IP Address	Database Name	Port # (default)	Username & Password
Oracle	Native	Required	Oracle database SID	Required (1521)	Required
	ODBC		ODBC DSN name		Required
	OCI		Database TNS name		Required
MySQL	Native	Required	MySQL database name	Optional (3306)	Optional
	ODBC		ODBC DSN name		Optional
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
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
Ingres	Native	Required	Database name		Required
	ODBC		ODBC DSN name		Required

Database	Access Method	Server Name or IP Address	Database Name	Port # (default)	Username & Password
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
Teradata	Native	Required	Database name		Required
	ODBC		ODBC DSN name		Required
Oracle RDB	Native	Required	Database name		Required
	ODBC		ODBC DSN name		Required
H2	Native	Required	Database name		Required
	ODBC		ODBC DSN name		Required
Netezza	Native	Required	Database name	Required (5480)	Required
	ODBC		ODBC DSN name		Required
IBM Universe	Native	Required	Database name		Required
	ODBC		ODBC DSN name		Required
SQLite	Native	Required	Database name		Required
	ODBC		ODBC DSN name		Required
Apache Derby	Native	optional	Database name	Optional (1527)	Optional
	ODBC		ODBC DSN name		Optional
Generic (*)	Native	Required	Database name	Required (Any)	Required
	ODBC		ODBC DSN name		Optional

(*) The generic database connection also needs to specify the URL and Driver class in the Generic tab! We now also allow these fields to be specified using a variable. That way you can access data from multiple database types using the same transformations and jobs. Make sure to use clean ANSI SQL that works on all used database types in that case.

5.6. MS SQL Server

To specify the SQL Server instance name, go to the Options tab and specify the parameter as shown:



	Database Type	Parameter	Value
1	MS SQL Server	instance	INSTANCE1

The SQL Server "instance" property

To enable single sign-on login you can specify in a similar fashion the ***domain*** option: From the jTDS FAQ on http://jtds.sourceforge.net/faq.html:

> *Specifies the Windows domain to authenticate in. If present and the user name and password are provided, jTDS uses Windows (NTLM) authentication instead of the usual SQL Server authentication (i.e. the user and password provided are the domain user and password). This allows non-Windows clients to log in to servers which are only configured to accept Windows authentication.*
>
> *If the* `domain` *parameter is present but no user name and password are provided, jTDS uses its native Single-Sign-On library and logs in with the logged Windows user's credentials (for this to work one would obviously need to be on Windows, logged into a domain, and also have the SSO library installed -- consult README.SSO in the distribution on how to do this).*

5.7. Oracle

This version of Pentaho Data Integration ships with the Oracle JDBC driver version 10.2.0. It is in general the most stable and recent driver we could find. However, if you do have issues with Oracle connectivity or other strange problems, you might want to consider replacing the 10.2. JDBC driver to match your database server. Replace files "ojdbc14.jar" and "orai18n.jar" in the directory libext/JDBC of your distribution with the files found in the $ORACLE_HOME/jdbc directory on your server.

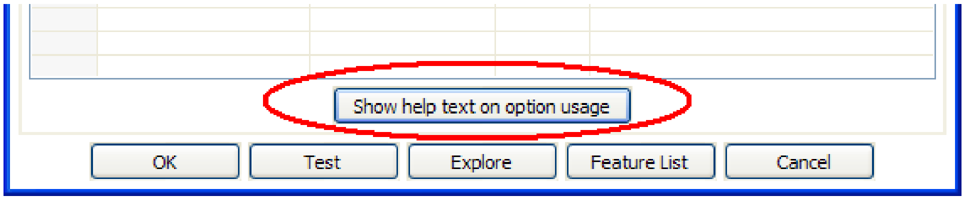
If you want to use OCI and an Oracle Net8 client, please read on. For OCI to work, the JDBC driver version used in Kettle needs to match your Oracle client version. Oracle 2.5.0 shipped with version 10.1, 2.5.0 ships with version 10.2.
You can either install that version of the Oracle client or (probably easier) change the JDBC driver in PDI if versions don't match up. (see above)

5.8. MySQL

Because by default, MySQL gives back complete query results in one block to the client (Kettle in this case) we had to enable "result streaming" by default. The big drawback of this is that it allows only 1 (one) single query to be opened at any given time. If you run into trouble because of that, you can disable this option in the MySQL tab of the database connection dialog.

Another issue you might come across is that the default timeout in the MySQL JDBC driver is set to 0. (no timeout) This leads to a problem in certain situations as it doesn't allow Kettle to detect a server crash or sudden network failure if it happens in the middle of a query or open database connection. This in turn leads to the infinite stalling of a transformation or job. To solve this, set the "connectTimeout" and "socketTimeout" parameters for MySQL in the Options tab. The value to be specified is in milliseconds: for a 2 minute timeout you would specify value 120000 (2 x 60 x 1000).

Also check out the other options on the linked MySQL help page. You can access this page using the supplied button.



5.9. JNDI

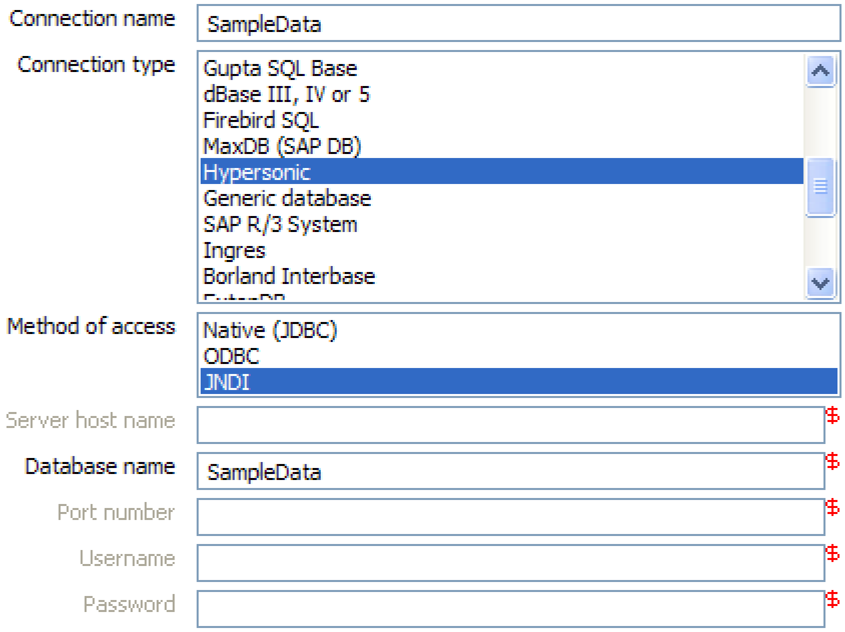
If you are developing transformations and jobs that will be deployed on an application server such as the Pentaho platform running on JBoss, you can configure your database connections using JNDI.

Because you don't want to have an application server running all the time during development or testing of the transformations, we have supplied a way of configuring a JNDI connection for “local” Kettle use.

To configure, edit properties file called “simple-jndi/jdbc.properties”

For example, to connect to the databases used in Pentaho Demo platform download, use this information in the properties file:

```
SampleData/type=javax.sql.DataSource
SampleData/driver=org.hsqldb.jdbcDriver
SampleData/url=jdbc:hsqldb:hsql://localhost/sampledata
SampleData/user=pentaho_user
SampleData/password=password
Quartz/type=javax.sql.DataSource
Quartz/driver=org.hsqldb.jdbcDriver
Quartz/url=jdbc:hsqldb:hsql://localhost/quartz
Quartz/user=pentaho_user
Quartz/password=password
Hibernate/type=javax.sql.DataSource
Hibernate/driver=org.hsqldb.jdbcDriver
Hibernate/url=jdbc:hsqldb:hsql://localhost/hibernate
Hibernate/user=hibuser
Hibernate/password=password
Shark/type=javax.sql.DataSource
Shark/driver=org.hsqldb.jdbcDriver
Shark/url=jdbc:hsqldb:hsql://localhost/shark
Shark/user=sa
Shark/password=
```



Connection name: SampleData

Connection type: Hypersonic

Method of access: JNDI

Server host name:

Database name: SampleData

Port number:

Username:

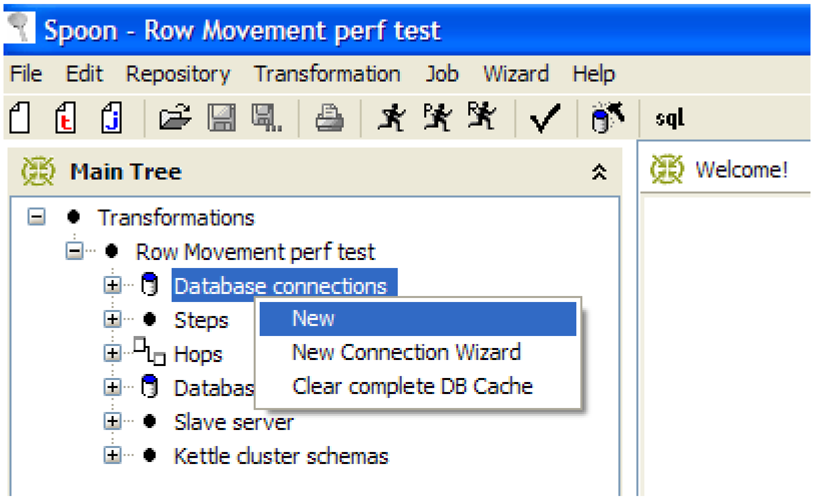
Password:

It is obviously important that the information stored in this file in the simple-jndi directory mirrors the content of your application server data sources.

5.10. Usage

5.10.1. Create a new connection

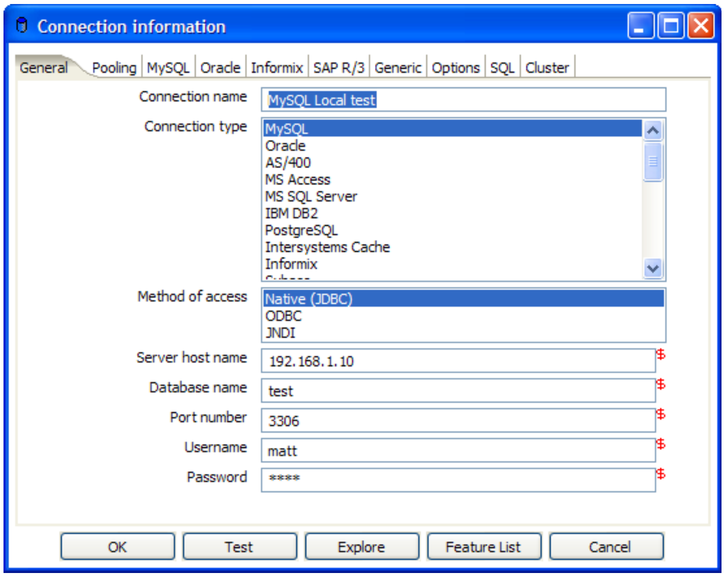
You can create a new connection by right clicking on the "Connections" tree entry and selecting "new".



Create new database connection - popup menu

Or : simply double click on the Connections tree entry.

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



Connection name: MySQL Local test

Connection type:
MySQL
Oracle
AS/400
MS Access
MS SQL Server
IBM DB2
PostgreSQL
Intersystems Cache
Informix

Method of access:
Native (JDBC)
ODBC
JNDI

Server host name: 192.168.1.10

Database name: test

Port number: 3306

Username: matt

Password: ****

Database connection dialog

5.10.2. General Tab

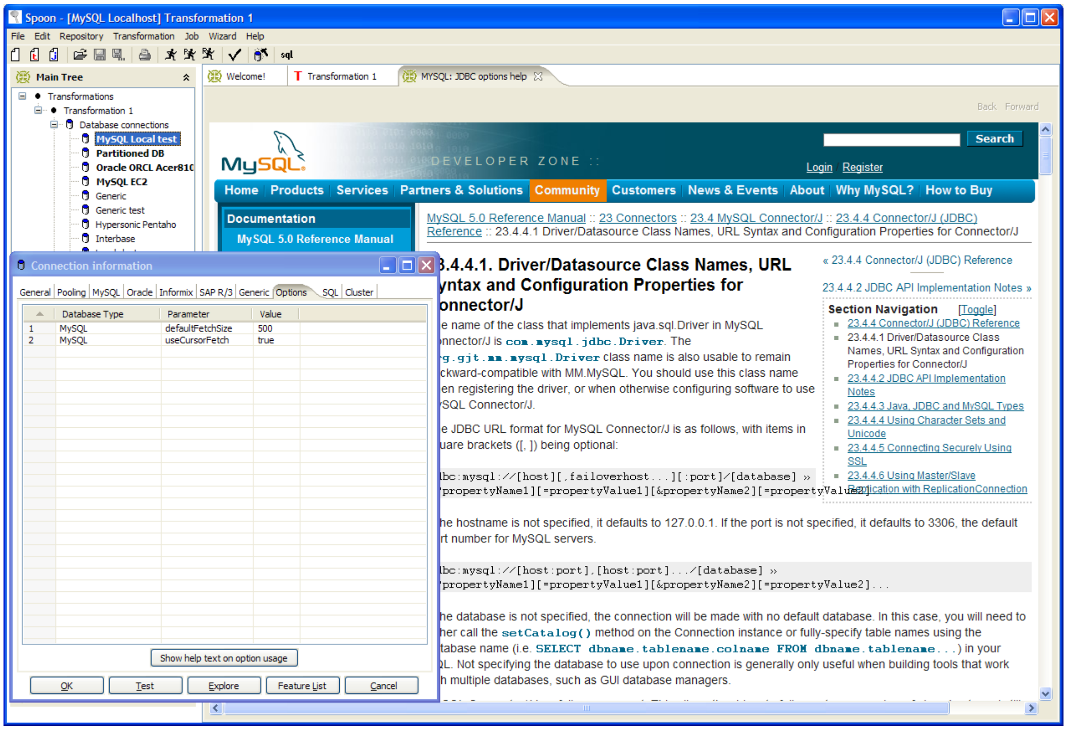
The general tab is where you setup the basic information about your connection like the connection name, type, access method, server name and login credentials.

5.10.2.1. Pooling Tab

The pooling tab allows you to configure your connection to use connection pooling and define options related to connection pooling like the initial pool size, maximum pool size and connection pool parameters.

5.10.2.2. Options Tab

This tab displays a variety of configurable, data source specific options (i.e. defaultFetchSize, useCursorFetch, etc.). Clicking on the 'Show help text on option usage' button will launch a browser window inside Spoon:


Spoon - [MySQL Localhost] Transformation 1
File Edit Repository Transformation Job Wizard Help
Main Tree
Transformations
Transformation 1
Database connections
MySQL Local test
Partitioned DB
Oracle ORCL Acer810
MySQL EC2
Generic
Generic test
Hypersonic Pentaho
Interbase
Welcome!
Transformation 1
MYSQL: JDBC options help
Back Forward
Search
DEVELOPER ZONE ::
Login Register
Home Products Services Partners & Solutions Community Customers News & Events About Why MySQL? How to Buy
Documentation
MySQL 5.0 Reference Manual
Connection information
General Pooling MySQL Oracle Informix SAP R/3 Generic Options SQL Cluster
Show help text on option usage
OK Test Explore Feature List Cancel

	Database Type	Parameter	Value
1	MySQL	defaultFetchSize	500
2	MySQL	useCursorFetch	true

MySQL 5.0 Reference Manual :: 23 Connectors :: 23.4 MySQL Connector/J :: 23.4.4 Connector/J (JDBC) Reference :: 23.4.4.1 Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J

3.4.4.1. Driver/Datasource Class Names, URL yntax and Configuration Properties for onnector/J

e name of the class that implements java.sql.Driver in MySQL onnector/J is `com.mysql.jdbc.Driver`. The `g.gjt.mm.mysql.Driver` class name is also usable to remain ckward-compatible with MM.MySQL. You should use this class name en registering the driver, or when otherwise configuring software to use SQL Connector/J.

e JDBC URL format for MySQL Connector/J is as follows, with items in uare brackets ([,]) being optional:

```
lbc:mysql://[host][,failoverhost...][:port]/[database] »
'propertyName1][=propertyValue1][&propertyName2][=propertyValue2]...
```

he hostname is not specified, it defaults to 127.0.0.1. If the port is not specified, it defaults to 3306, the default rt number for MySQL servers.

```
lbc:mysql://[host:port],[host:port].../[database] »
'propertyName1][=propertyValue1][&propertyName2][=propertyValue2]...
```

he database is not specified, the connection will be made with no default database. In this case, you will need to her call the `setCatalog()` method on the Connection instance or fully-specify table names using the tabase name (i.e. `SELECT dbname.tablename.colname FROM dbname.tablename...`) in your L. Not specifying the database to use upon connection is generally only useful when building tools that work h multiple databases, such as GUI database managers.

« 23.4.4 Connector/J (JDBC) Reference

23.4.4.2 JDBC API Implementation Notes »

Section Navigation [Toggle]

- 23.4.4 Connector/J (JDBC) Reference
- 23.4.4.1 Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J
- 23.4.4.2 JDBC API Implementation Notes
- 23.4.4.3 Java, JDBC and MySQL Types
- 23.4.4.4 Using Character Sets and Unicode
- 23.4.4.5 Connecting Securely Using SSL
- 23.4.4.6 Using Master/Slave Replication with ReplicationConnection

Display options help in a Spoon browser

5.10.2.3. SQL Tab

This tab allows you to enter a number of SQL commands immediately after connecting to the database. This is sometimes needed for various reasons like licensing, configuration, logging, tracing, etc.

5.10.2.4. Cluster Tab

This tab allows you to enable clustering for the database connection and create connections to the data partitions. To create a new data partition, enter a partition ID and the hostname, port, database, username and password for connecting to the partition.

5.10.3. Edit a connection

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

5.10.4. Duplicate a connection

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

5.10.5. Copy to clipboard

Copies the XML describing the connection to the clipboard.

5.10.6. Delete a connection

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

5.10.7. 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.

5.10.8. Execute SQL commands on a connection

Right click on the connection name and select "SQL Editor". See also 5. SQL editor for more information.

5.10.9. 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.

5.10.10. Explore

This option will start the database explorer for the selected database connection. Please see 4.5.9. Database Explorer for more information.

5.11. 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 or JDBC connection.

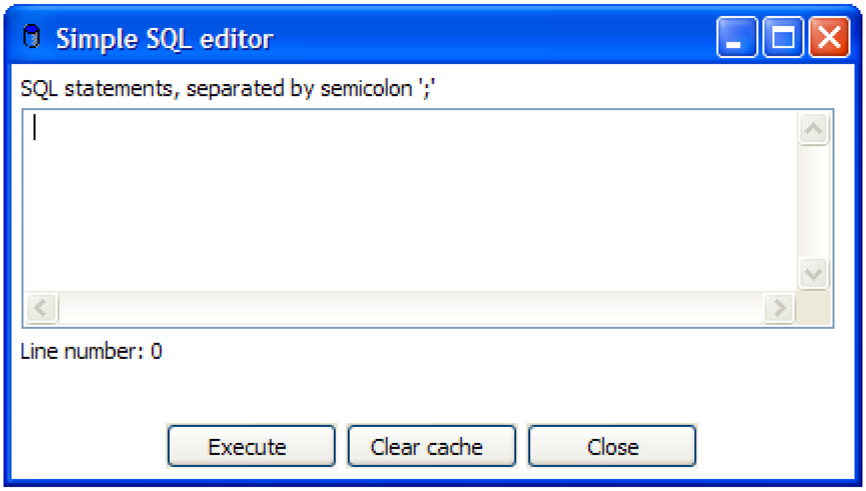
6. SQL Editor

6.1. 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.*

6.2. Screen shot



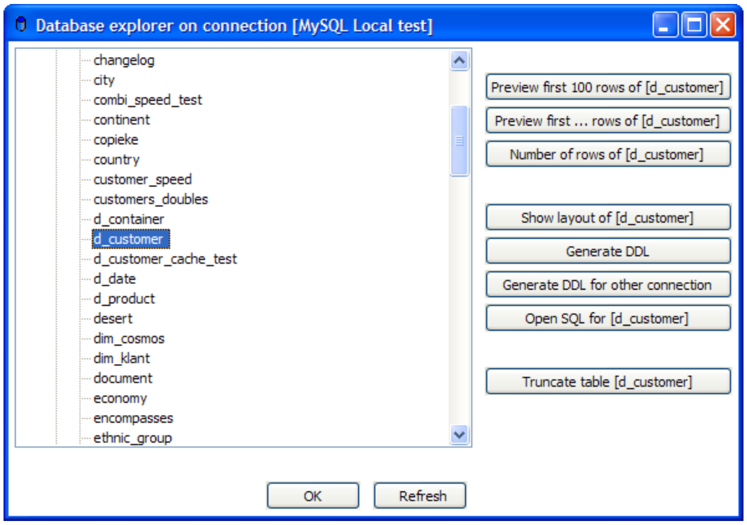
The Simple SQL Editor dialog

6.3. Limitations

This is a simple SQL editor. It doesn't know all the dialects of all the more than 20 supported databases. That means that creating stored procedures, triggers and other database specific objects might pose problems. Please consider using the tools that came with the database in that case.

7. Database Explorer

7.1. Screen shot


Database explorer on connection [MySQL Local test]
changelog
city
combi_speed_test
continent
copieke
country
customer_speed
customers_doubles
d_container
d_customer
d_customer_cache_test
d_date
d_product
desert
dim_cosmos
dim_klant
document
economy
encompasses
ethnic_group
Preview first 100 rows of [d_customer]
Preview first ... rows of [d_customer]
Number of rows of [d_customer]
Show layout of [d_customer]
Generate DDL
Generate DDL for other connection
Open SQL for [d_customer]
Truncate table [d_customer]
OK
Refresh

The database explorer dialog

7.2. 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 use the buttons on the right or to right click on a shown table or view at the lowest level in the tree. You can then 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)
- Truncate table ... : generates a *commented out* "Truncate table" statement.

8. Hops

8.1. 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).

8.1.1. Transformation Hops

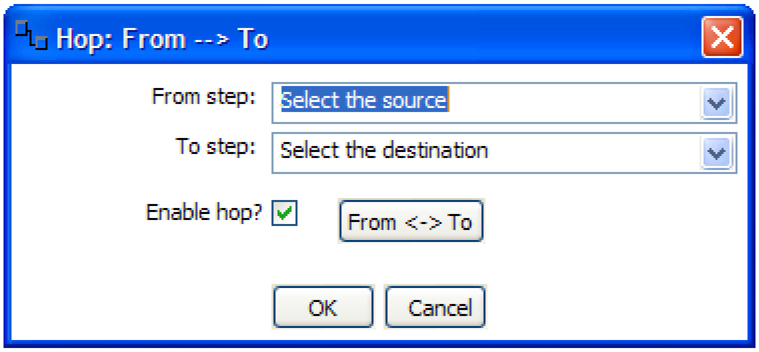
When a hop is disabled in a transformation, the steps downstream of the disabled hop are cut off from any data flowing upstream of the disabled hop. This may lead to unexpected results when editing the downstream steps. For example, if a particular step-type offers a "Get Fields" button, clicking the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

8.1.2. Job Hops

Besides the execution order, it also specifies the condition on which the next job entry will be executed. You can specify the evaluation mode by right clicking on the job hop:

- "Unconditional" specifies that the next job entry will be executed regardless of the result of the originating job entry.
- "Follow when result is true" specifies that the next job entry will only be executed when the result of the originating job entry was true, meaning successful execution, file found, table found, without error, evaluation was false, ...
- "Follow when result is false" specifies that the next job entry will only be executed when the result of the originating job entry was false, meaning unsuccessful execution, file not found, table not found, error(s) occurred, evaluation was false, ...

8.2. Screen shots

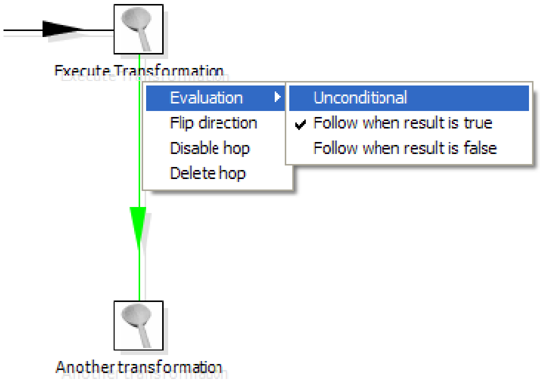

Hop: From --> To

From step: Select the source

To step: Select the destination

Enable hop? ☑

Editing a Transformation Hop



Editing a Job Hop

8.3. 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"

8.4. 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.

8.5. Loops

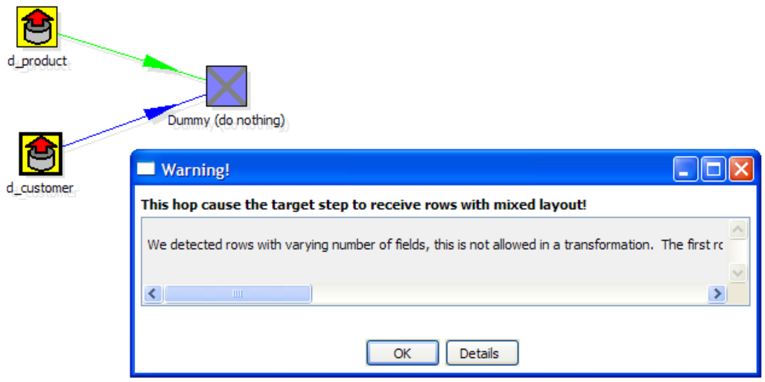
Loops are not allowed in transformations because Spoon depends heavily on the previous steps to determine the field values that are passed from one step to another. If we would allow loops in transformations we often would get endless loops and undetermined results.

Loops **are** allowed in jobs because Chef executes job entries sequentially. Just make sure
you don't build endless loops. Please check out Evaluation job entry (12.2.10. Evaluation).
This job entry can help you exit closed loops based on the number of times a job entry was executed.

8.6. Mixing rows: trap detector

Mixing rows with different layout is not allowed in a transformation. However, the developers still receive many bug reports when in fact people are mixing rows of data. This is causing steps to fail because fields can't be found where expected or the data type changes unexpectedly.

For that reason we added a "trap detector" when you by accident mix rows:



This hop cause the target step to receive rows with mixed layout!

We detected rows with varying number of fields, this is not allowed in a transformation. The first rc

In this case the full eror report reads*: We detected rows with varying number of fields, this is not allowed in a transformation. The first row contained 13 fields, another one contained 16 : [customer_tk=0, version=0, date_from=, date_to=, CUSTOMERNR=0, NAME=, FIRSTNAME=, LANGUAGE=, GENDER=, STREET=, HOUSNR=, BUSNR=, ZIPCODE=, LOCATION=, COUNTRY=, DATE_OF_BIRTH=]*

Please note that this is a warning and will not prevent you from performing the task you want to do.

8.7. Transformation hop colors

Where the color coding of the job hops are fairly simple, transformation hops can have all kinds of appearances.
Here is an overview:

Appearance	Meaning
Green	Distributes rows
Red	Copies rows
Yellow	Provides info for step, distributes rows
Magenta	Provides info for step, copies rows
Gray	Disabled
Black	Named target step
Blue	Candidate hop using middle button + drag
Orange (Dot line)	Always unused because no data is ever going to go there
Red (Bold Dot line)	Carrying rows that caused errors in source steps

9. Variables

9.1. Variable usage

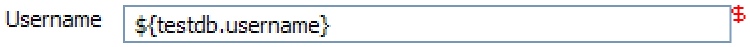
Variables can be used throughout Pentaho Data Integration, including transformation steps and job entries. Variables can be defined by setting them with the "Set Variable" step in a transformation or by setting them in the `kettle.properties` file in the directory:

$HOME/.kettle (Unix/Linux/OSX)
C:\Documents and Settings\<username>\.kettle\ (Windows)

The way to use them is either by grabbing them using step 11.4.45. 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. Dialogs that support variable usage throughout Pentaho Data Integration are visually indicated using a red dollar sign like this:



Username ${testdb.username} $

Property allowing variable usage

You can use the CTRL-SPACE keyboard to select a variable to be inserted into the property value. Mouse over the $ sign to see a shortcut help text displayed.

9.2. Variable scope

The scope of a variable is defined by the place in which it is defined.

9.2.1. Environment variables

The first usage (and only usage in previous Kettle versions) was to set an environment variable. This was traditionally done by passing options to the Java Virtual Machine (JVM) with the -D option. It's also an easy way to specify the location of temporary files in a platform independent way, for example using variable ${java.io.tmpdir}. This variable points to directory /tmp on Unix/Linux/OSX and to C:\Documents and Settings\<username\Local Settings\Temp on Windows machines.

The only problem with using environment variables is that the usage is not dynamic in nature and problems arise if you would try to use them in a dynamic way. That is because if you run 2 or more transformations or jobs run at the same time on an application server (for example the Pentaho framework) you would get conflicts. Changes to the environment variables are visible to all software running on the virtual machine.

9.2.2. Kettle variables

Because the scope of an environment variable (9.2.1.Environment variables) is too broad, we introduced Kettle variables that are local to the job in which the variable is set.

The "Set Variable" step in a transformation allows you to specify in which job you want to set the variable. (parent job, grand-parent job or the root job)

9.2.3. Internal variables

These variables are always defined:

Variable Name	Sample value
Internal.Kettle.Build.Date	2007/05/22 18:01:39
Internal.Kettle.Build.Version	2045
Internal.Kettle.Version	2.5.0

These variables are defined in a transformation:

Variable Name	Sample value
Internal.Transformation.Filename.Directory	D:\Kettle\samples
Internal.Transformation.Filename.Name	Denormaliser - 2 series of key-value pairs.ktr
Internal.Transformation.Name	Denormaliser - 2 series of key-value pairs sample
Internal.Transformation.Repository.Directory	/

These are the internal variables that are defined in a Job:

Variable Name	Sample value
Internal.Job.Filename.Directory	/home/matt/jobs
Internal.Job.Filename.Name	Nested jobs.kjb
Internal.Job.Name	Nested job test case
Internal.Job.Repository.Directory	/

These variables are defined in a transformation running on a slave server, executed in clustered mode:

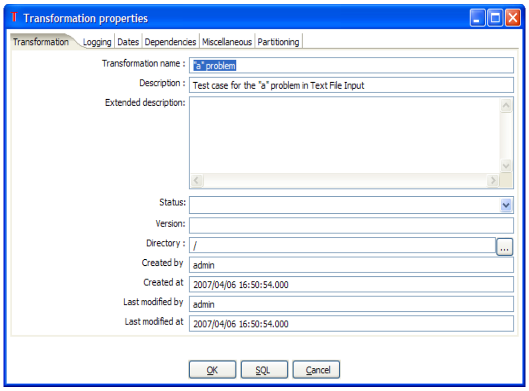
Variable Name	Sample value
Internal.Slave.Transformation.Number	0..<cluster size-1> (0,1,2,3 or 4)
Internal.Cluster.Size	<cluster size> (5)

10. Transformation Settings

10.1. Description

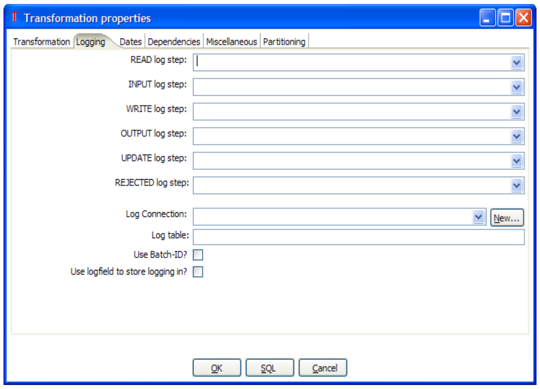
There are a few options that control how a transformation is behaving and how it is logging what it is doing. Open the Transformation Settings from the main menu under Transformation|Settings

10.2. Screen shots

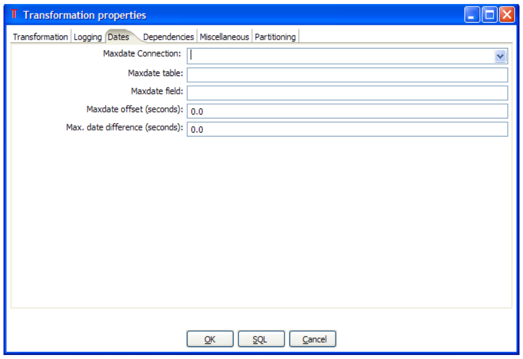


Transformation name : "a" problem
Description : Test case for the "a" problem in Text File Input
Directory : /
Created by admin
Created at 2007/04/06 16:50:54.000
Last modified by admin
Last modified at 2007/04/06 16:50:54.000

The "Transformation" tab

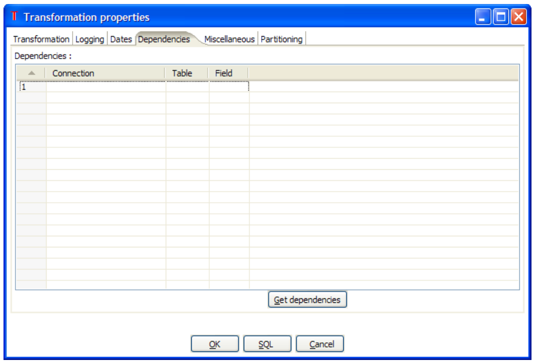


The "Logging" tab

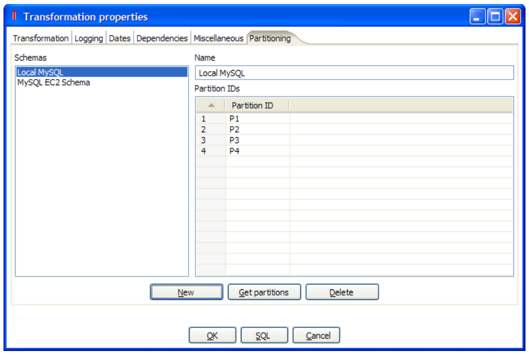


Maxdate offset (seconds): 0.0
Max. date difference (seconds): 0.0

The "Dates" tab



The "Dependencies" tab

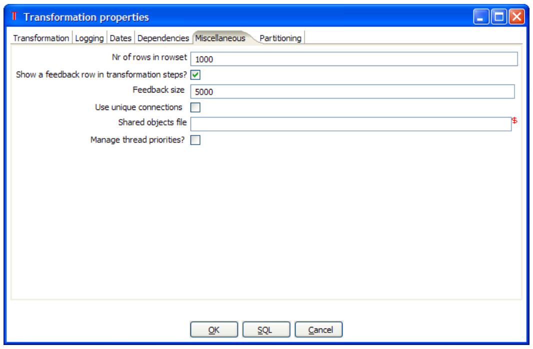


Name: Local MySQL

Partition IDs

	Partition ID
1	P1
2	P2
3	P3
4	P4

The "Partitioning" Tab



Nr of rows in rowset 1000
Show a feedback row in transformation steps? ☑
Feedback size 5000
Use unique connections □
Manage thread priorities? □

The "Miscellaneous" tab

10.3. Options

Option	Description
Transformation name	the name of the transformation. Required information if you want to save to a repository.
Description	Short description of the transformation, shown in the repository explorer
Extended description	Long extended description of the transformation
Status	Draft or production status
Version	Version description
Directory	the directory in the repository in which you want to save the transformation.
Created by	Displays the repository user name of the user that created the transformation.
Last modified at	Displays the date and time when the transformation was created.
Last modified by	Displays the user name of the last user that modified the transformation.
Last modified at	Displays the date and time when the transformation was last modified.
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

Option	Description
	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.
Show a feedback row in transformation steps?	This controls whether or not to add a feedback entry into the log file while the transformation is being executed. By default, this feature is enabled and configured to display a feedback record every 5000 rows.
The feedback size	Sets the number of rows to process before entering a feedback entry into the log. Set this higher when processing large amounts of data to reduce the amount of information in the log file.
Use unique connections	This allows use to open one unique connection per defined and used database connection in the transformation. Checking this option is required in order to allow a failing transformation to be rolled back completely.
Shared objects file	Specifies the location of the XML file used to stored shared objects like database connections, clustering schemas and more.
Manage thread priorities?	Allows you to enable or disable the internal logic for changing the Java thread priorities based on the number of input and output rows in the perspective "rowset" buffers. This can be useful in some simplistic situations where the cost of using the logic is exceeds the benefit of the thread prioritization.
Schemas	Provides a list of available database partitions. Create a new partition by clicking on the "New" button. The "Get Partitions" button will retrieve a list of available partitions that have been defined for the connection. For more information on setting up database partitions, see 4. Database Connections.

10.4. 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.

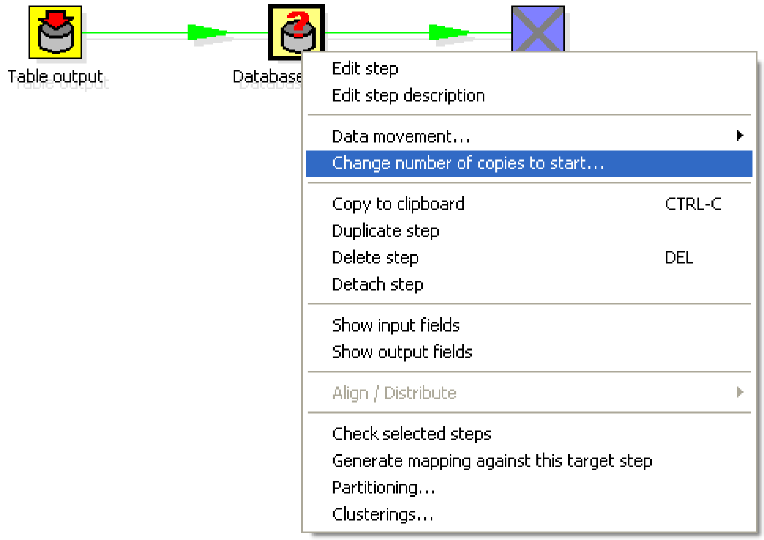
11. Transformation Steps

11.1. 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 (10.4. Step Types) for a complete list of all available step-types.

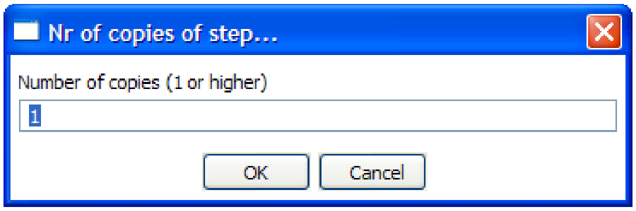
11.2. Launching several copies of a step

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…”



The "Step copies" popup menu

You will get this dialog:

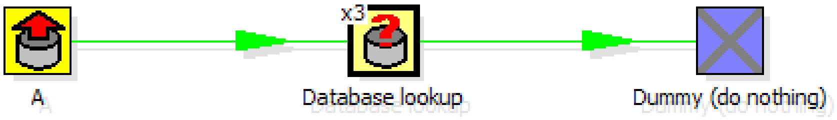


Number of copies (1 or higher)

1

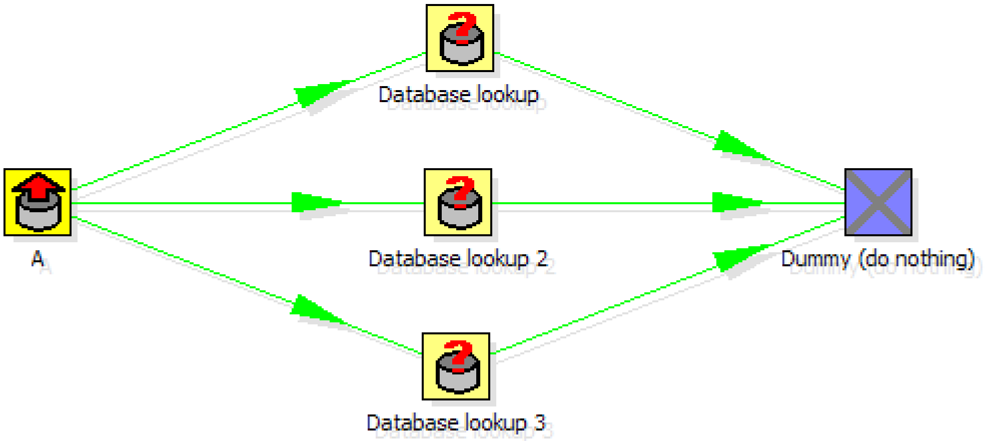
The step copies dialog

If you enter 3 this will be shown:



Multiple step copies example

It is technically the equivalent of this:

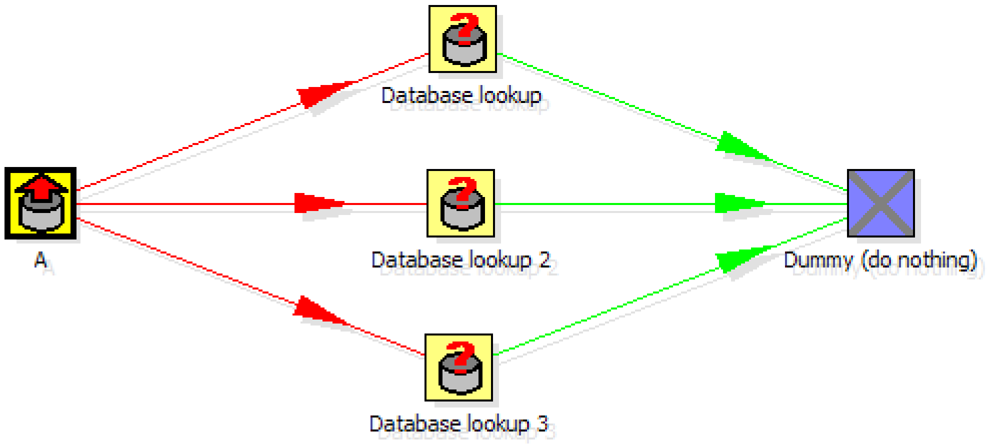


Multiple step copies equivalent

11.3. 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 rows transformation sample

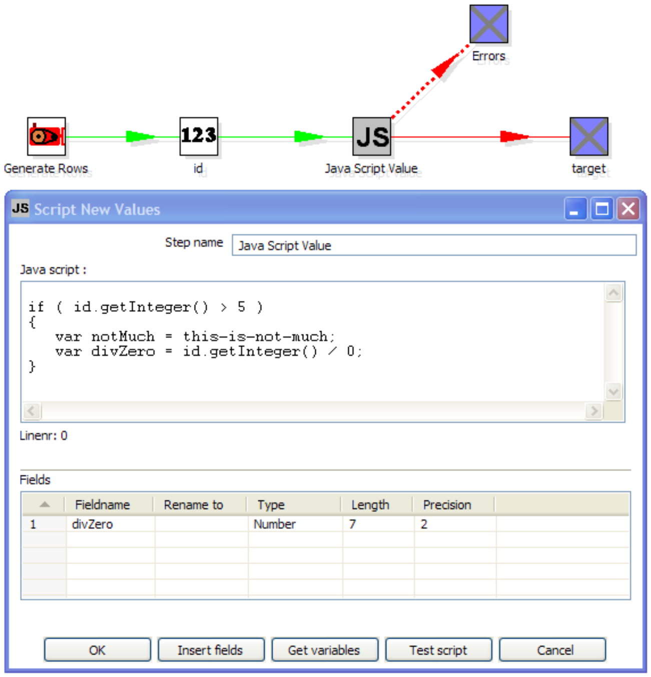
"Copy data" means that all rows from step "A" are copied to all 3 the target steps.
In this case it means that step "B" gets 3 copies of all the rows that "A" has sent out.

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".*

11.4. Generic error handling

This feature originated as a great idea from Sven Boden, one of the core developers of Pentaho Data Integration. The idea was simple: in stead of halting a transformation when an error occurs in a step, you should be able to pass those rows that cause an error to a different step.

In the example below we artificially generate an error in the Script Values step when an ID is higher than 5.

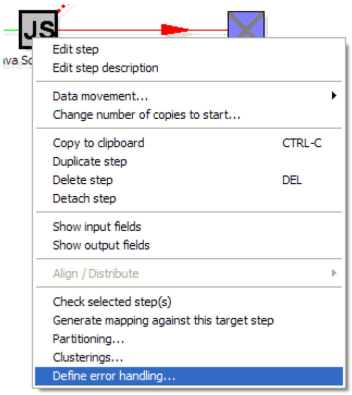


Step name: Java Script Value

```
if ( id.getInteger() > 5 )
{
   var notMuch = this-is-not-much;
   var divZero = id.getInteger() / 0;
}
```

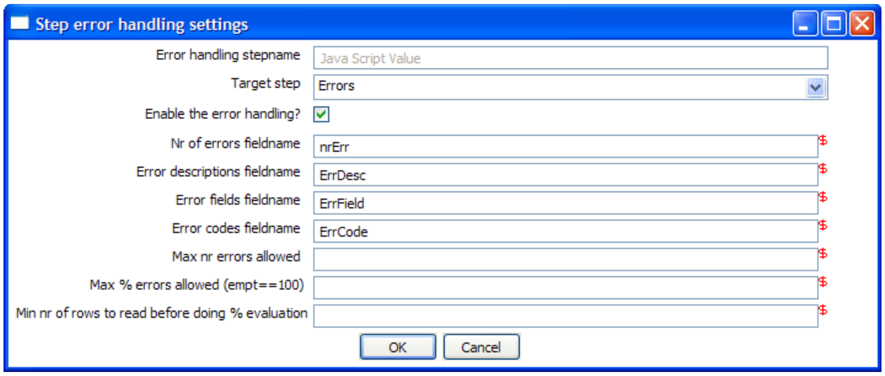
	Fieldname	Rename to	Type	Length	Precision
1	divZero		Number	7	2

To configure the error handling, you can right click on the step involved and select the "Error handing..." menu item:



NOTE: this menu item only appears when clicking on steps that support the new error handling code.

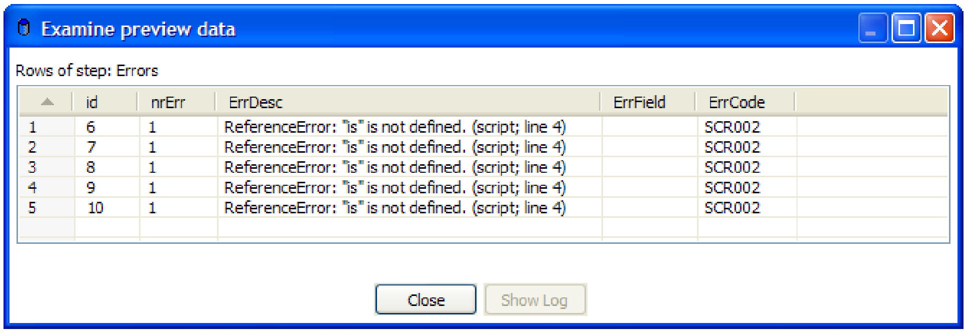
The error handling dialog looks like this:



Step error handling settings

Error handling stepname	Java Script Value
Target step	Errors
Enable the error handling?	☑
Nr of errors fieldname	nrErr
Error descriptions fieldname	ErrDesc
Error fields fieldname	ErrField
Error codes fieldname	ErrCode
Max nr errors allowed	
Max % errors allowed (empt==100)	
Min nr of rows to read before doing % evaluation	

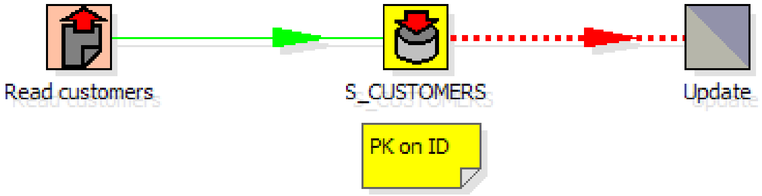
As you can see, you can add extra fields being to the "error rows":



Rows of step: Errors

	id	nrErr	ErrDesc	ErrField	ErrCode
1	6	1	ReferenceError: "is" is not defined. (script; line 4)		SCR002
2	7	1	ReferenceError: "is" is not defined. (script; line 4)		SCR002
3	8	1	ReferenceError: "is" is not defined. (script; line 4)		SCR002
4	9	1	ReferenceError: "is" is not defined. (script; line 4)		SCR002
5	10	1	ReferenceError: "is" is not defined. (script; line 4)		SCR002

This way, we can easily define new data flows in our transformations. The typical use-case for this is an alternative way of doing an Upsert (Insert/Update):



This transformation performs an insert regardless of the content of the table. If you put a primary key on the ID (in this case the customer ID) the insert into the table cause an error. Because of the error handling we can pass the rows in error to the update step. Preliminary tests have shown this strategy of doing upserts to be 3 times faster in certain situations. (with a low updates to inserts ratio)

11.5. Apache Virtual File System (VFS) support

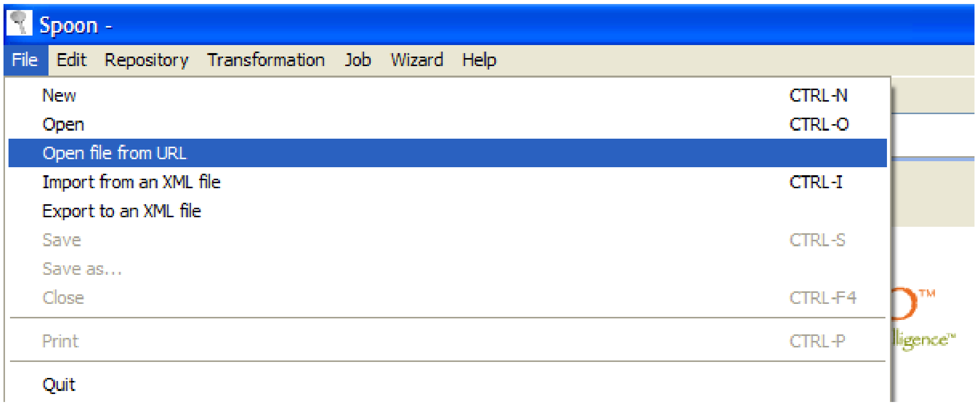
One of the new things that we implemented in version 2.5.0 is the ability to reference source files, transformations and jobs from any location you like.

The underlying libraries we use to do that is the Apache Commons Virtual File System.

Here is a simple example that you can try out yourself:

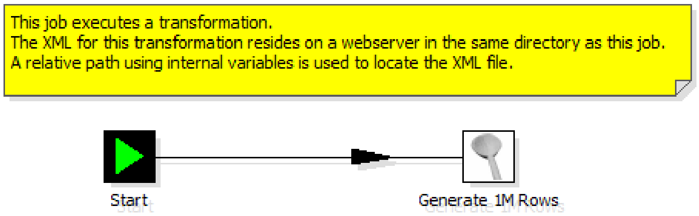
```
sh kitchen.sh -file:http://www.kettle.be/GenerateRows.kjb
```

Let’s have a look at this job in Spoon. To open it directly from the URL above follow this procedure:



Type in the URL:

Selecting OK will then load the job in Spoon:

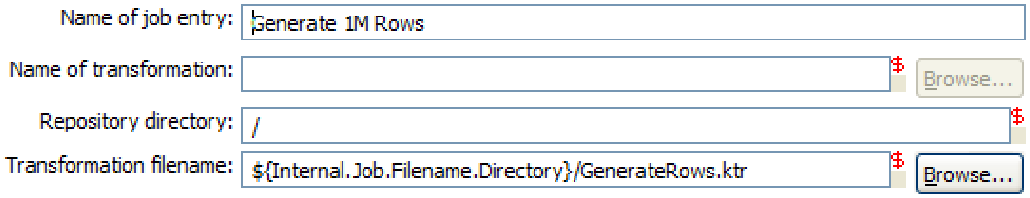


This job executes a transformation.
The XML for this transformation resides on a webserver in the same directory as this job.
A relative path using internal variables is used to locate the XML file.

The transformation we are about to launch is also located on the webserver. The internal variable for the job name directory is:

Internal.Job.Filename.Directory   http://www.kettle.be/

This allows us to reference the transformation as follows:



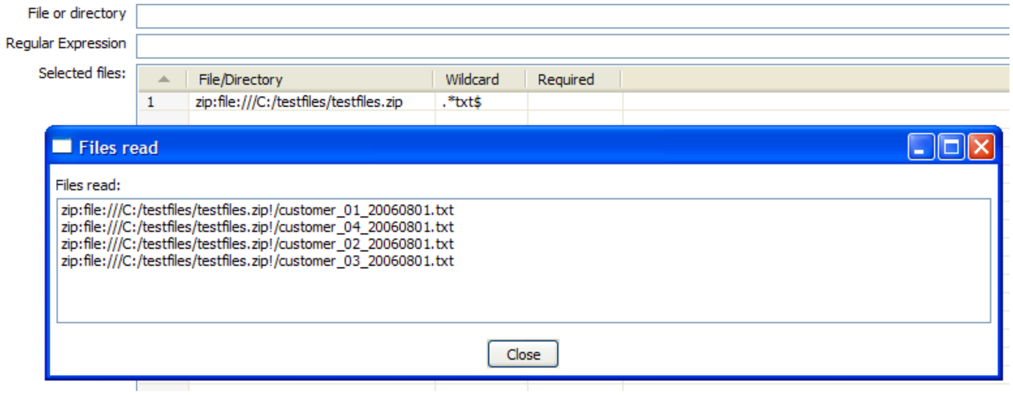
Name of job entry:	Generate 1M Rows
Name of transformation:	
Repository directory:	/
Transformation filename:	${Internal.Job.Filename.Directory}/GenerateRows.ktr

Please note that if you try this yourself you'll note that you can't save the job back to the webserver. That is not because we don't support that, but because you don't have the permission to so.

Please have a quick look at the almost endless list of possibilities over here. These include direct loading from zip-files, gz-files, jar-files, ram drives, SMB, (s)ftp, (s)http, etc.

We will extend this list even further in the near future with our own drivers for the Pentaho solutions repository and later on for the Kettle repository (something like: psr:// and pdi:// URIs)

Below is another nice VFS example:



	File/Directory	Wildcard	Required
1	zip:file:///C:/testfiles/testfiles.zip	.*txt$	

Files read:

```
zip:file:///C:/testfiles/testfiles.zip!/customer_01_20060801.txt
zip:file:///C:/testfiles/testfiles.zip!/customer_04_20060801.txt
zip:file:///C:/testfiles/testfiles.zip!/customer_02_20060801.txt
zip:file:///C:/testfiles/testfiles.zip!/customer_03_20060801.txt
```

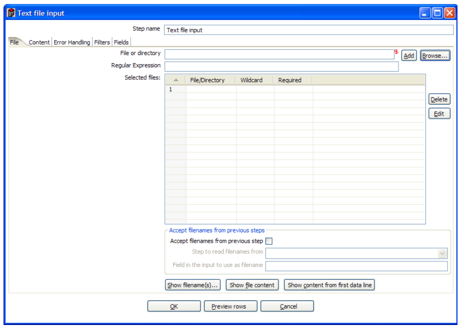
As you can see, you can use a wild-card to directly select files inside of a zip file.

Apache VFS support was implemented in all steps and job entries that are part of the Pentaho Data Integration suite as well as in the recent Pentaho platform code and in Pentaho Analyses (Mondrian).

11.6. Transformation Step Types

11.6.1. Text File Input

11.6.1.1. Screen shots


Text file input

Step name: Text file input

File | Content | Error Handling | Filters | Fields

File or directory:

Regular Expression:

Selected files:

	File/Directory	Wildcard	Required
1			

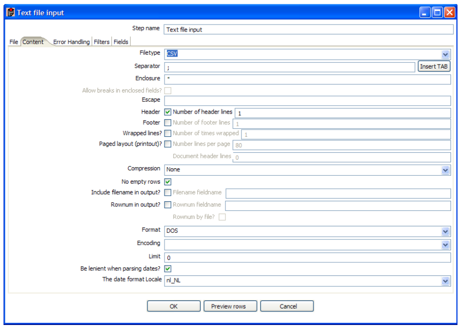
Accept filenames from previous steps

Accept filenames from previous step ☐

Step to read filenames from:

Field in the input to use as filename:

Text File Input "File" tab


Text file input

Step name: Text file input

File | Content | Error Handling | Filters | Fields

Filetype: CSV

Separator: ;

Enclosure: "

Allow breaks in enclosed fields? ☐

Escape:

Header ☑ Number of header lines: 1

Footer ☐ Number of footer lines: 1

Wrapped lines? ☐ Number of times wrapped: 1

Paged layout (printout)? ☐ Number lines per page: 80

Document header lines: 0

Compression: None

No empty rows ☑

Include filename in output? ☐ Filename fieldname:

Rownum in output? ☐ Rownum fieldname:

Rownum by file? ☐

Format: DOS

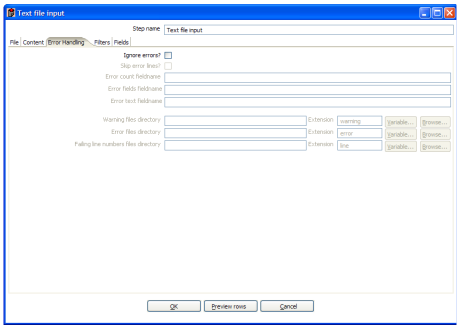
Encoding:

Limit: 0

Be lenient when parsing dates? ☑

The date format Locale: nl_NL

Text File Input "Content" tab


Text file input

Step name: Text file input

File | Content | Error Handling | Filters | Fields

Ignore errors? ☐

Skip error lines? ☐

Error count fieldname:

Error fields fieldname:

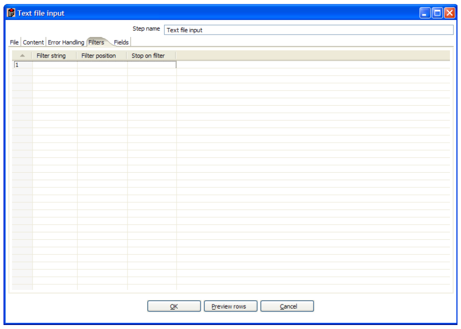
Error text fieldname:

Warning files directory: Extension: warning

Error files directory: Extension: error

Failing line numbers files directory: Extension: line

Text File Input "Error handling" tab

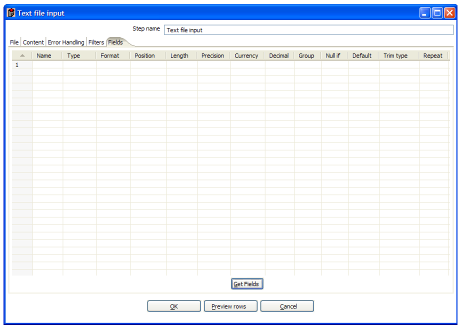

Text file input

Step name: Text file input

File | Content | Error Handling | Filters | Fields

	Filter string	Filter position	Stop on filter
1			

Text File Input "Filter" tab


Text file input

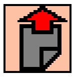
Step name: Text file input

File | Content | Error Handling | Filters | Fields

	Name	Type	Format	Position	Length	Precision	Currency	Decimal	Group	Null if	Default	Trim type	Repeat
1													

Text File Input "Fields" tab

11.6.1.2. Icon



11.6.1.3. General description

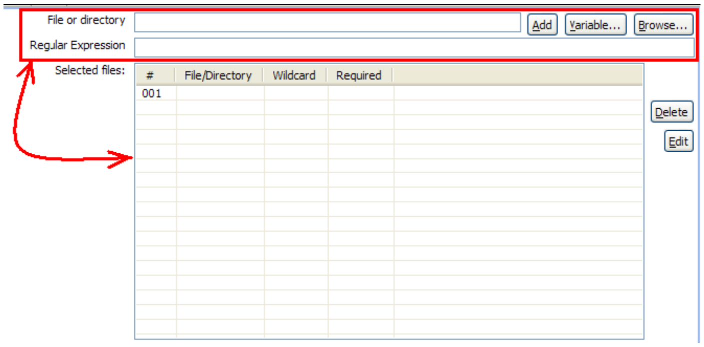
You can use the Text File Input step to read a large number of different text-files.

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 wild cards 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.

11.6.1.4. Options

11.6.1.4.1. 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:



Adding entries to the list of files

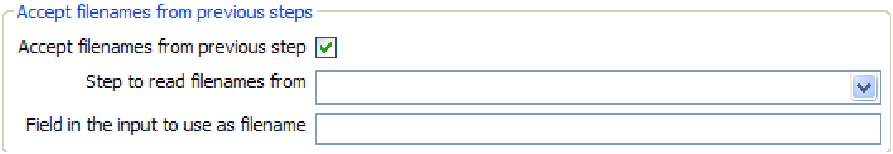
You can also have this step search for files by specifying a wild card in the form of a regular expression. Regular expressions are more sophisticated than simply using '*' and '?' wild cards.

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. NOTE: 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.

11.6.1.4.2. Accept filenames from previous step



Accept filenames from previous steps

Accept filenames from previous step ☑

Step to read filenames from

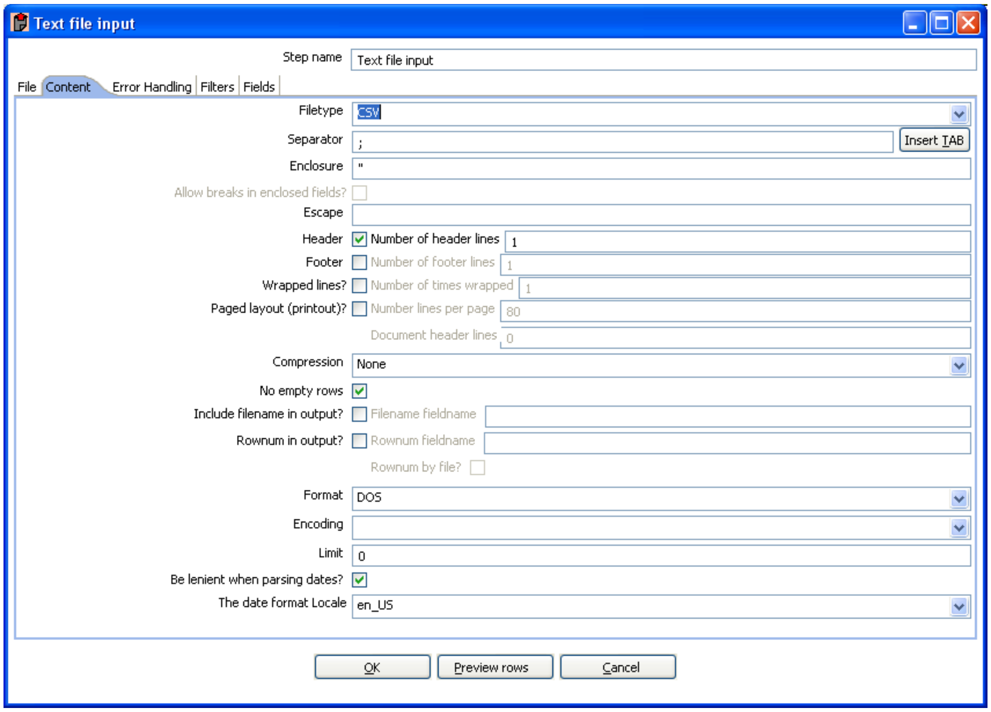
Field in the input to use as filename

Accepting filenames from previous steps

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 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.

11.6.1.4.3. Content specification



Step name: Text file input

Filetype: CSV

Separator: ;

Enclosure: "

Allow breaks in enclosed fields? □

Escape:

Header ☑ Number of header lines 1

Footer □ Number of footer lines 1

Wrapped lines? □ Number of times wrapped 1

Paged layout (printout)? □ Number lines per page 80

Document header lines 0

Compression: None

No empty rows ☑

Include filename in output? □ Filename fieldname

Rownum in output? □ Rownum fieldname

Rownum by file? □

Format: DOS

Encoding:

Limit: 0

Be lenient when parsing dates? ☑

The date format Locale: en_US

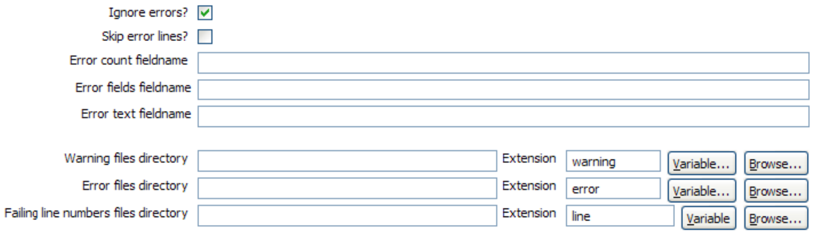
The content tab allows you to specify the format of the text files that are being read. Here is a list of the options on this tab:

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.
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 ***data*** 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.

Option	Description
Compression	Enable this option if your text file is placed in a Zip or GZip archive. **NOTE**: *At the moment, only the first file in the archive is read.*
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.
Rownum 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.
Rownum by file?	Allows the row number to be reset per file.
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 and 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 32nd will become Feb 1st.
The date format Locale	This locale is used to parse dates that have been written in full like “February 2nd, 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.

11.6.1.4.4. 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: warning

Error files directory — Extension: error

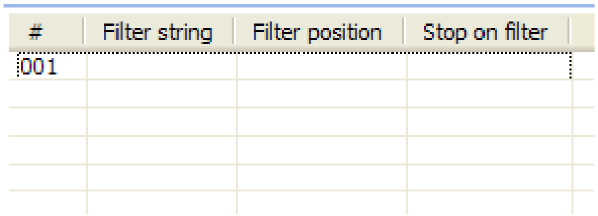
Failing line numbers files directory — Extension: line

Specifiying the error handling

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>

11.6.1.4.5. 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			

Specifying text file filters

Option	Description
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.

11.6.1.4.6. *Fields*

Option	Description
Name	name of the field
Type	Type of the field can be either String, Date or Number
Format	See 10.5.1.5. Formats for a complete description of format symbols.
Length	For *Number*: Total number of significant figures in a number; For *String:* total length of string; For *Date:* length of printed output of the string (e.g. 4 only gives back the year).
Precision	For *Number:* Number of floating point digits; For *String, Date, Boolean:* 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

11.6.1.5. Formats

11.6.1.5.1. *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

Symbol	Location	Localized	Meaning
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".

11.6.1.5.2. 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

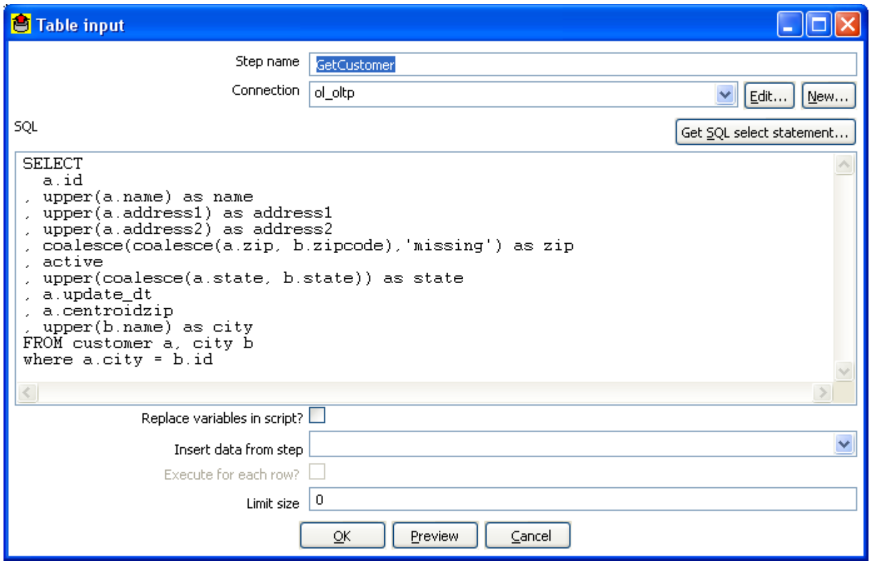
Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm 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

11.6.1.6. Extras

Function/Button	Description
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.

11.6.2. Table input

11.6.2.1. Screenshot



Step name: GetCustomer

Connection: ol_oltp

```
SELECT
   a.id
,  upper(a.name) as name
,  upper(a.address1) as address1
,  upper(a.address2) as address2
,  coalesce(coalesce(a.zip, b.zipcode),'missing') as zip
,  active
,  upper(coalesce(a.state, b.state)) as state
,  a.update_dt
,  a.centroidzip
,  upper(b.name) as city
FROM customer a, city b
where a.city = b.id
```

Replace variables in script? □

Insert data from step

Execute for each row? □

Limit size 0

11.6.2.2. Icon

11.6.2.3. General description

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

11.6.2.4. 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.

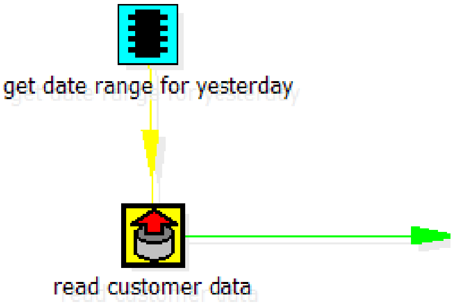
11.6.2.5. Example

Consider for example the following SQL statement:

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

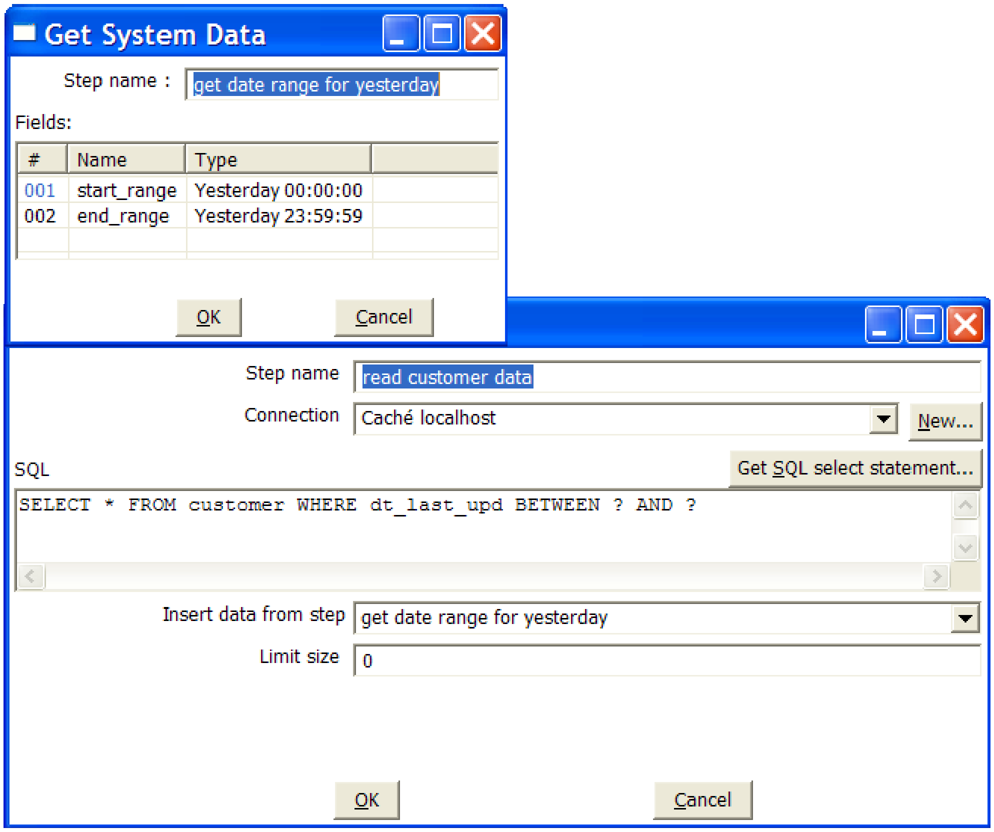
This statement needs 2 dates that are read on the "Insert data from" step.

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:

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

Step name: read customer data

Connection: Caché localhost

SQL

```
SELECT * FROM customer WHERE dt_last_upd BETWEEN ? AND ?
```

Insert data from step: get date range for yesterday

Limit size: 0

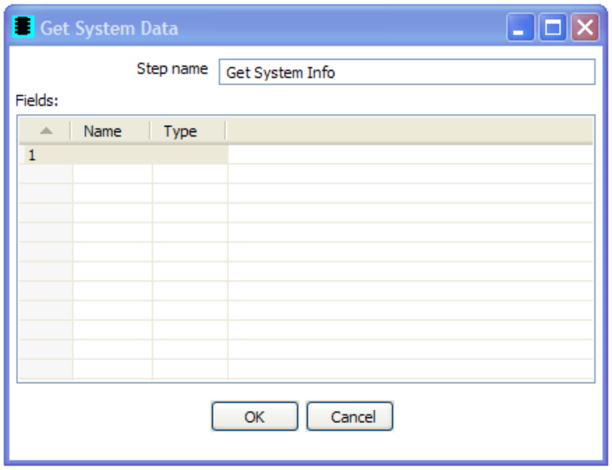
See also: Get System Data step 7.4.16. Get System Info.

11.6.2.6. Extras

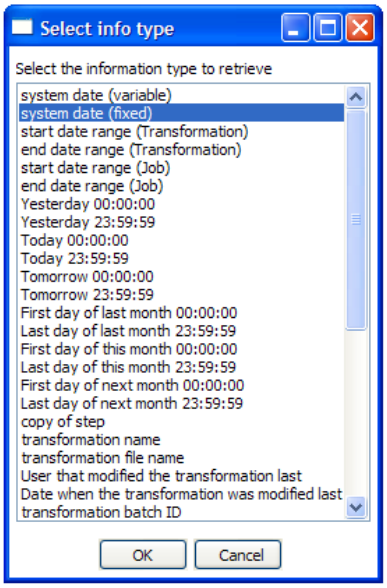
Function/Button	Description
Preview	This option previews this step. It is done by preview of a new transformation with 2 steps: this one and a Dummy step. You can see the detailed logging of that execution by clicking on the logs button in the the preview window.

11.6.3. Get System Info

11.6.3.1. Screenshot



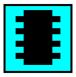
Step name: Get System Info



Select the information type to retrieve

- system date (variable)
- system date (fixed)
- start date range (Transformation)
- end date range (Transformation)
- start date range (Job)
- end date range (Job)
- Yesterday 00:00:00
- Yesterday 23:59:59
- Today 00:00:00
- Today 23:59:59
- Tomorrow 00:00:00
- Tomorrow 23:59:59
- First day of last month 00:00:00
- Last day of last month 23:59:59
- First day of this month 00:00:00
- Last day of this month 23:59:59
- First day of next month 00:00:00
- Last day of next month 23:59:59
- copy of step
- transformation name
- transformation file name
- User that modified the transformation last
- Date when the transformation was modified last
- transformation batch ID

11.6.3.2. Icon



11.6.3.3. 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 (Transformation)	Start of date range, based upon information in ETL log table. See, also 9. Transformation settings
end date range (Transformation)	End of date range, based upon information in ETL log table. See, also 9. Transformation settings
start data range (Job)	Start of date range based upon information in the ETL log table. See also 11. Job settings
End date range (Job)	End of date range based upon information in the ETL log table. See also 11. Job settings

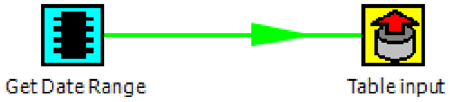
Option	Description
Yesterday 00:00:00	Start of yesterday.
Yesterday 23:59:59	End of yesterday.
Today 00:00:00	Start of today.
Today 23:59:59	End of today.
Tomorrow 00:00:00	Start of tomorrow.
Tomorrow 23:59:59	End of tomorrow
First day of last month 00:00:00	Start of last month.
Last day of last month 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.
copy of step	Copy nr of the step (see also: 7.2. Launching several copies of a step).
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 6. Transformation settings.
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.
Kettle version	Returns the Kettle version (e.g. 2.5.0)
Kettle Build Version	Returns the build version of the core Kettle library (e.g. 13)
Kettle Build Date	Returns the build date of the core Kettle library

11.6.3.4. Options

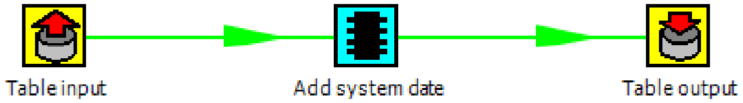
- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **Fields**: The fields to output.

11.6.3.5. Usage

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

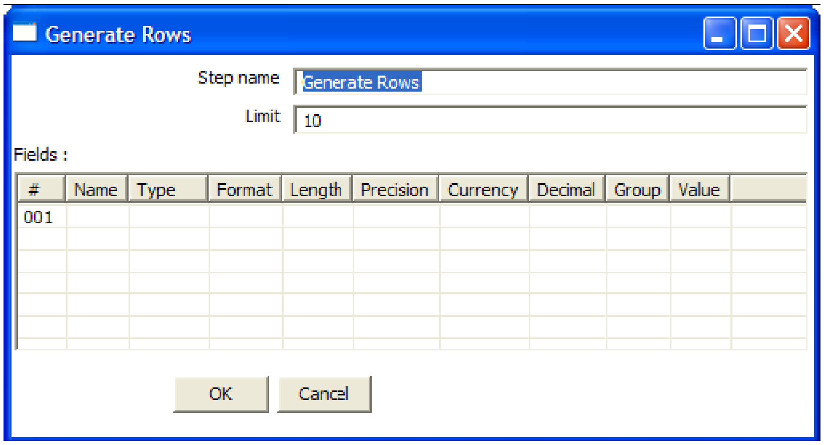


From version 2.3.0 on, this step also accepts input rows. The selected values will be added to the rows found in the input stream(s):



11.6.4. Generate Rows

11.6.4.1. Screenshot



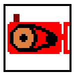
Step name: Generate Rows

Limit: 10

Fields :

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value
001									

11.6.4.2. Icon



11.6.4.3. General description

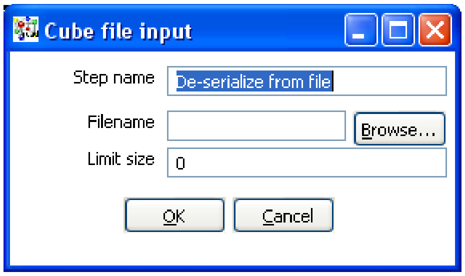
This step type outputs a number of rows, default empty but optionally containing a number of static fields.

11.6.4.4. Options

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

11.6.5. De-serialize from file (formerly Cube Input)

11.6.5.1. Screenshot

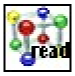


Step name: De-serialize from file

Filename:

Limit size: 0

11.6.5.2. Icon



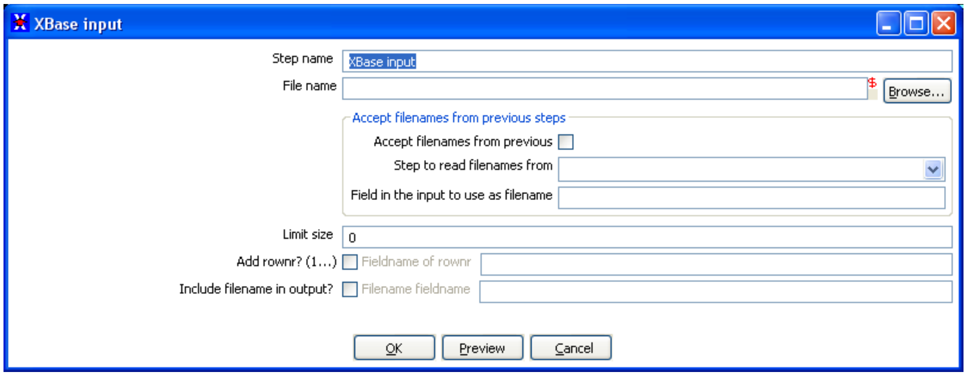
11.6.5.3. General description

Read rows of data from a binary Kettle cube file. (see also "Serialize to file")

NOTE: ***This step should only be used to store short lived data. It is not guaranteed that the file format stays the same between versions of Pentaho Data Integration.***

11.6.6. XBase input

11.6.6.1. Screenshot



XBase input

Step name: XBase input

File name:

Accept filenames from previous steps

Accept filenames from previous □

Step to read filenames from:

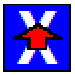
Field in the input to use as filename:

Limit size: 0

Add rownr? (1...) □ Fieldname of rownr:

Include filename in output? □ Filename fieldname:

11.6.6.2. Icon



11.6.6.3. General description

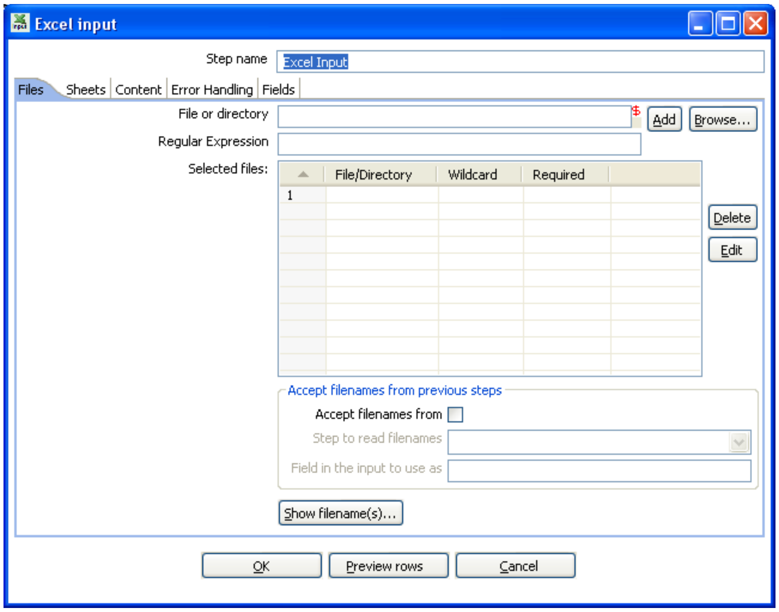
With this step it is possible to read data from most types of DBF file derivates called the XBase family. (dBase III/IV, Foxpro, Clipper, ...)

11.6.6.4. 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.

11.6.7. Excel input

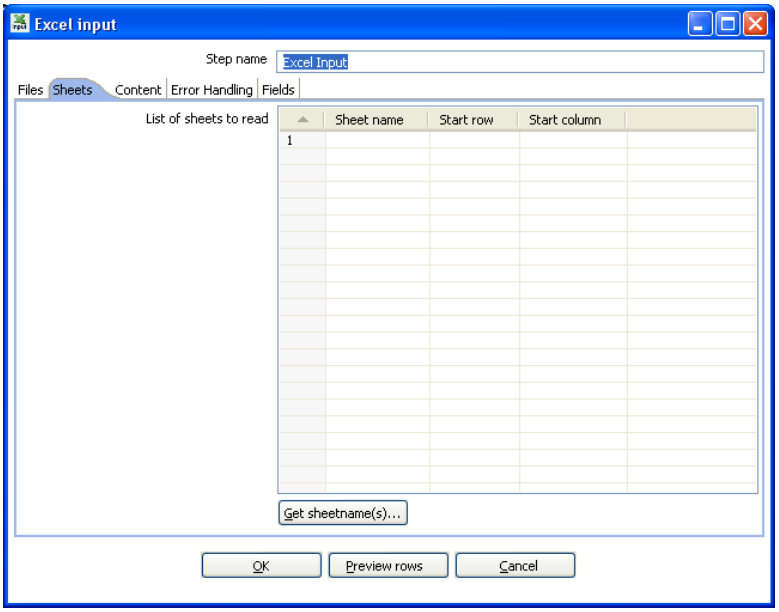
11.6.7.1. Screenshots



Step name: Excel Input

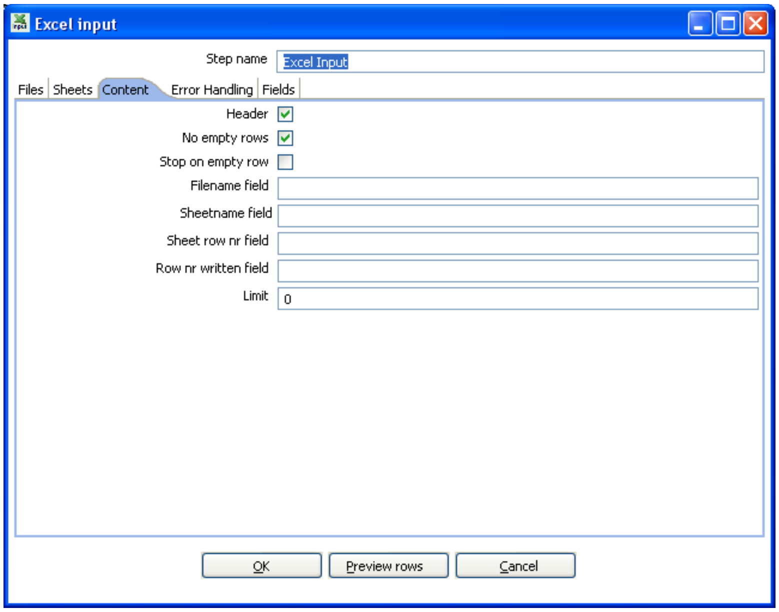
Accept filenames from ☐

The "Files" tab



Step name: Excel Input

The "Sheets" tab



Step name: Excel Input

Header ☑

No empty rows ☑

Stop on empty row □

Filename field

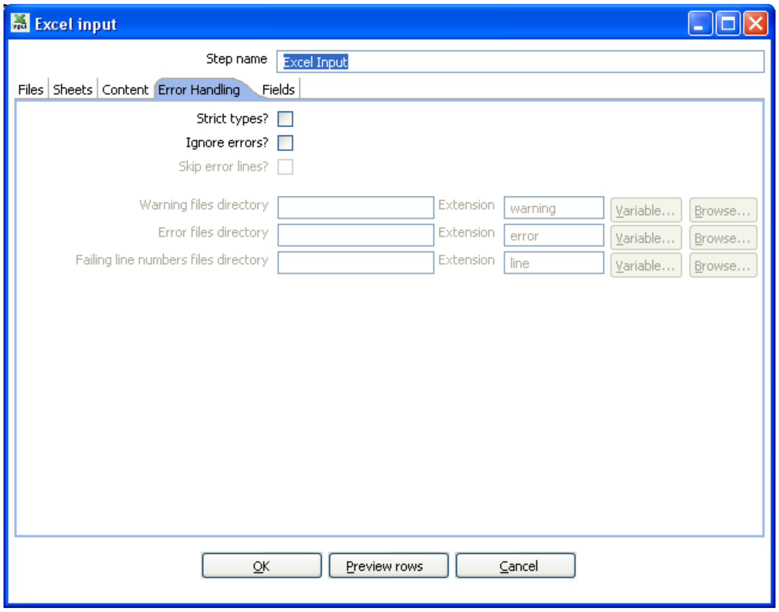
Sheetname field

Sheet row nr field

Row nr written field

Limit: 0

The "Content" tab



Step name: Excel Input

Strict types? □

Ignore errors? □

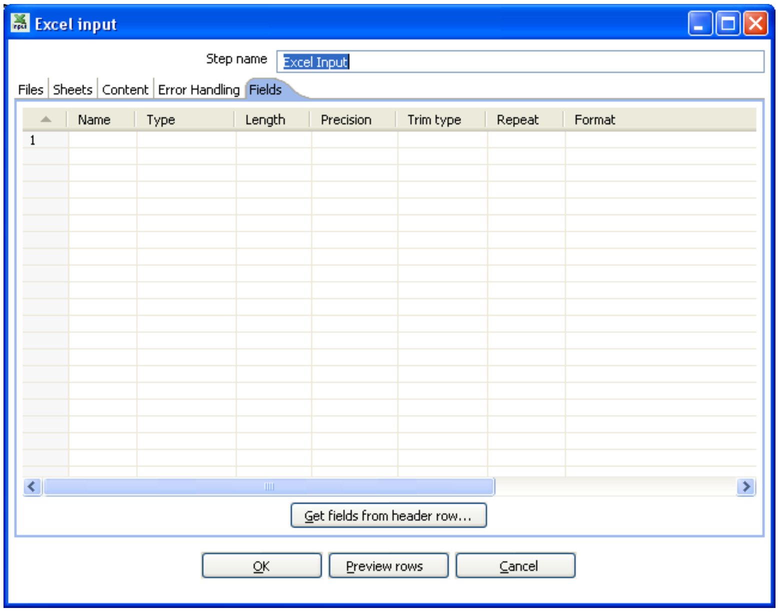
Skip error lines? □

Warning files directory — Extension: warning

Error files directory — Extension: error

Failing line numbers files directory — Extension: line

The "Error Handling" tab



Step name: Excel Input

	Name	Type	Length	Precision	Trim type	Repeat	Format
1							

The "Fields" tab

11.6.7.2. Icon

11.6.7.3. 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.

11.6.7.4. 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: 11.4.1.4.1.Filename specification options
- Please also see: 11.4.1.4.2.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.
- **Sheet row nr field**: Specify a field name to include the sheet row number in the output of the step. The sheet row number is the actual row number in the Excel sheet.
- **Row nr written field**: Specify a field name to include the row number in the output of the step. "Row number written" is the number of rows processed, starting at 1 and counting up regardless of sheets and files.
- **Limit**: limit the number of rows to this number, 0 means: all rows.

Error handling tab:

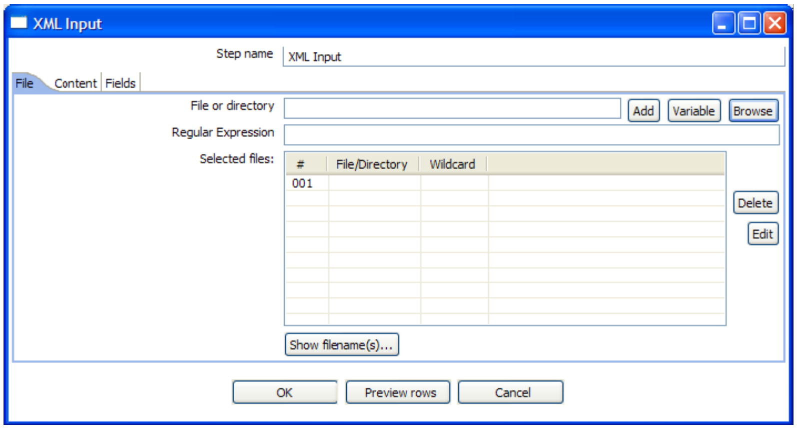
- Please see 11.4.1.4.4.Error handling for a detailed description on error handling.

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 28th, 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.

11.6.8. XML Input

11.6.8.1. Screenshot



Step name: XML Input

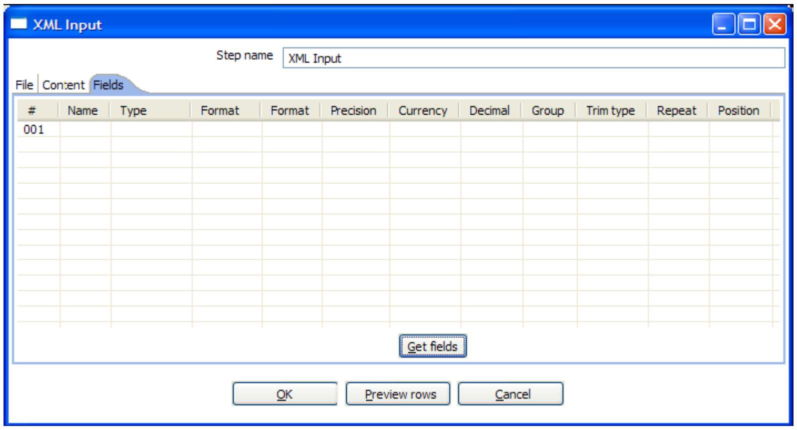
File or directory:

Regular Expression:

Selected files:

#	File/Directory	Wildcard
001		

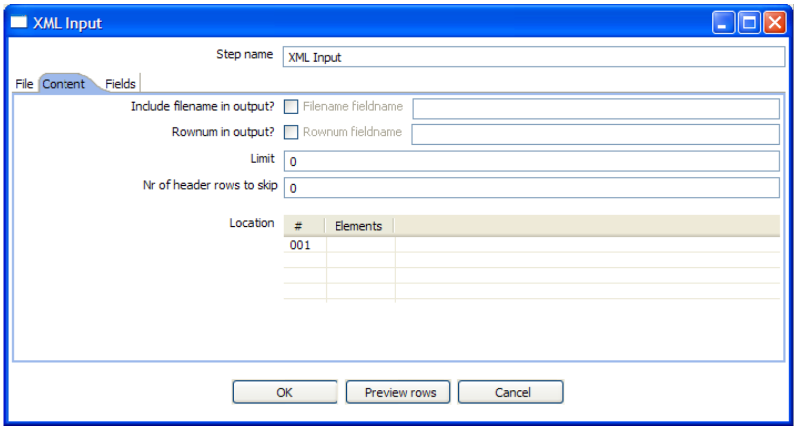
The "files" tab



Step name: XML Input

#	Name	Type	Format	Format	Precision	Currency	Decimal	Group	Trim type	Repeat	Position
001											

The "Content" tab



Step name: XML Input

Include filename in output? □ Filename fieldname

Rownum in output? □ Rownum fieldname

Limit: 0

Nr of header rows to skip: 0

Location:

#	Elements
001	

The "Fields" tab

11.6.8.2. Icon

11.6.8.3. 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.

11.6.8.4. 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. **NOTE**: *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.
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.

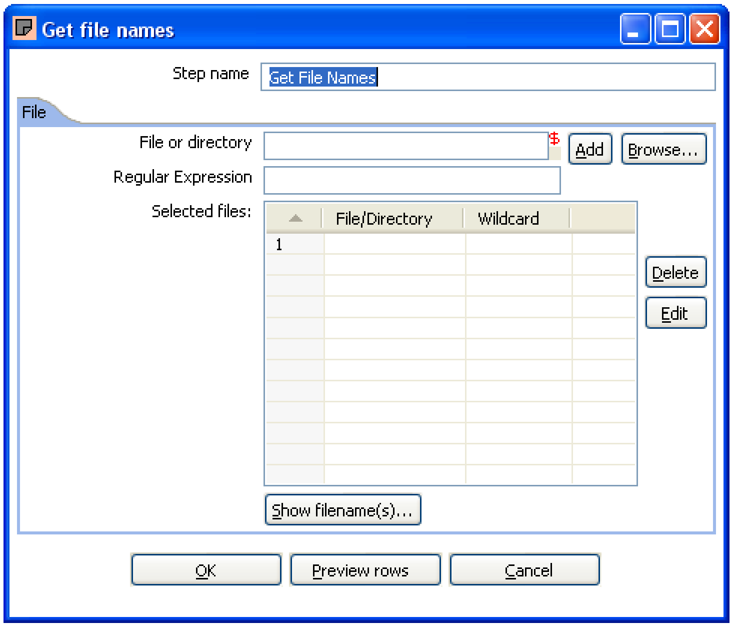
Location	Specify the path by way of elements to the repeating part of the XML file.
	For example if you are reading rows from this XML file:
	`<Rows>`
	`  <Row>`
	`    <Field1>...</Field1> ...`
	`  </Row>`
	`  ...`
	`</Rows>`
	Then you set the location to Rows, Row
	Note: *you can also set the root (Rows) as a repeating element location. The output will then contain 1 (one) row.*

Fields	• Name: Name of the field. • Type: Type of the field can be either String, Date or Number. • Format mask to convert with: See 8.5.1.5 for a complete description of format specifiers. • Length: ○ Number: Total number of significant figures in a number;

Option	Description
	○ String: total length of string; ○ Date: length of printed output of the string (e.g. 4 only gives back the year). • Precision: ○ Number: Number of floating point digits; ○ String: unused; ○ Date: unused. • Currency: Symbol used to represent currencies 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 "," (10,000.00) or "." (5.000,00) • Trim type: The trimming method to apply on the string found in the XML. • Repeat: Check this if you want to repeat empty values with the corresponding value from the previous row. • Position: The position of the XML element or attribute. You use the following syntax to specify the position of an element: The first element called “element”: `E=element/1` The first attribute called “attribute”: `A=attribute/1` The first attribute called “attribute” in the second “element” tag: `E=element/2, A=attribute/1` **NOTE**: *You can auto-generate all the possible positions in the XML file supplied by using the “Get Fields” button.* **NOTE**: *Support was added for XML documents where all the information is stored in the Repeating (or Root) element. The special R= locater was added to allow you to grab this information. The “Get fields” button finds this information if it's present.*

11.6.9. Get File Names

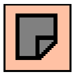
11.6.9.1. Screenshot



Step name: Get File Names

	File/Directory	Wildcard
1		

11.6.9.2. Icon



11.6.9.3. General description

This step allows you to get information regarding filenames on the file system on your system. Please also see 11.4.1.4.1.Filename specification options for a detailed description on how to use the specify filenames.

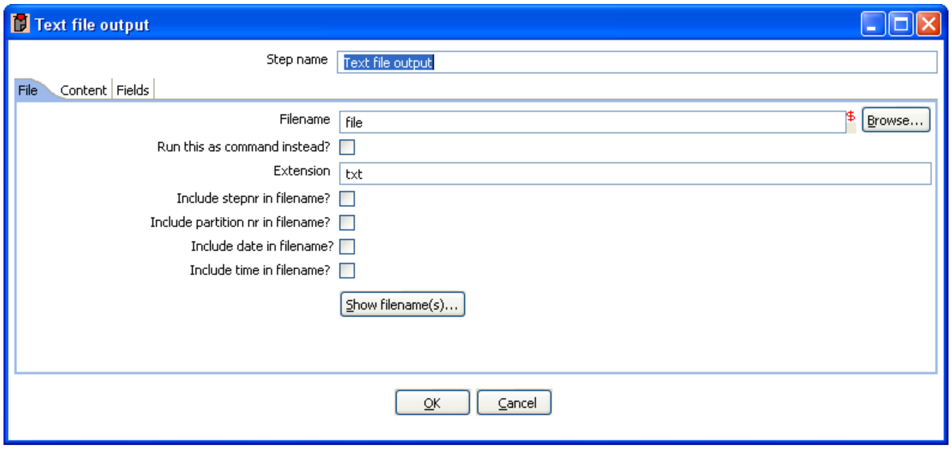
The output fields for this step are:

- 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/)

11.6.10. Text File Output

11.6.10.1. Screenshots

The "File" tab:



Text file output

Step name: Text file output

Filename: file

Run this as command instead? □

Extension: txt

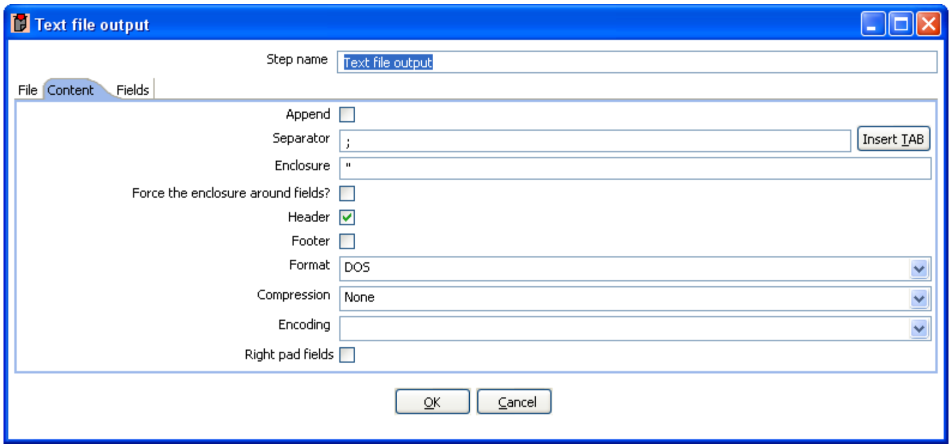
Include stepnr in filename? □

Include partition nr in filename? □

Include date in filename? □

Include time in filename? □

The "Content" tab:



Text file output

Step name: Text file output

Append □

Separator: ;

Enclosure: "

Force the enclosure around fields? □

Header ☑

Footer □

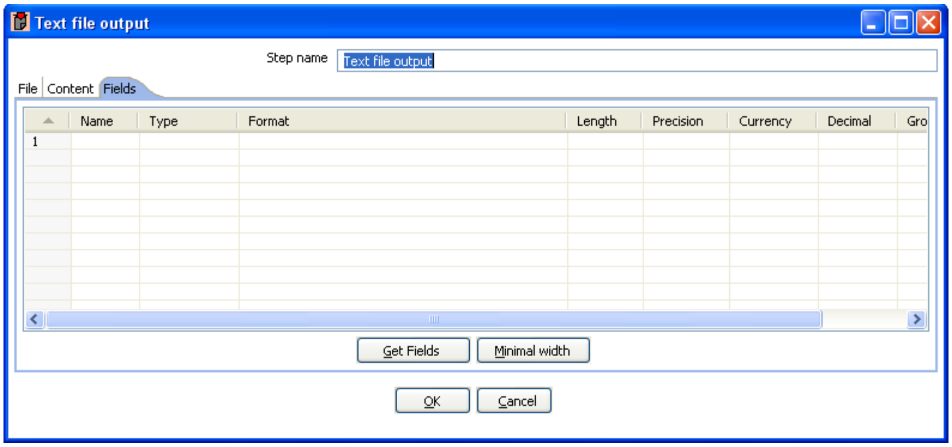
Format: DOS

Compression: None

Encoding:

Right pad fields □

The "Fields" tab:

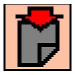


Text file output

Step name: Text file output

	Name	Type	Format	Length	Precision	Currency	Decimal	Gro
1								

11.6.10.2. Icon



11.6.10.3. 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.

11.6.10.4. 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.
Run this as a command instead?	Check this to "pipe" the results into the command or script you specify.
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 10.2. Launching several copies of a step), the copy number is included in the filename, before the extension. (_0).
Include partition nr in filename?	Includes the data partition number in the filename.
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).
Force the enclosure around fields?	This option forces all field names to be enclosed with the character specified in the Enclosure property above.
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).
Format	This can be either DOS or UNIX. UNIX files have lines are separated by linefeeds. DOS files have lines separated by carriage returns and 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.
Compression	Allows you to specify the type of compression, .zip or .gzip to use when compressing the output. **NOTE**: *At the moment, only one file is placed in a single archive.*
Right pad fields	Add spaces to the end of the fields (or remove characters at the end) until they have the specified length.
Fast data dump (no	Improves the performance when dumping large amounts of data to a text file

Option	Description
formatting)	by not including any formatting information.
Split every … rows	If this number N is larger than zero, split the resulting text-file into multiple parts of N rows.
Add Ending line of file	Allows you to specify an alternate ending row to the output file.
Fields	• Name: name of the field • Type: Type of the field can be either String, Date or Number • Format: See 10.4.1.5. Formats for a complete description of format specifiers • Length: ○ Number: Total number of significant figures in a number ○ String: total length of string; ○ Date: length of printed output of the string (e.g. 4 only gives back the year). • Precision: ○ Number: Number of floating point digits; ○ String: unused; ○ Date: unused. • Currency: Symbol used to represent currencies 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 "," (10,000.00) or "." (5.000,00) • Null: If the value of the field is null, insert this string into the textfile

11.6.10.5. Extras

Function/Button	Description
Show filenames	This option shows a list of the files the will be generated. NOTE: This is a simulation and sometimes depends on the number of rows in each file, etc.
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.

11.6.11. Table output

11.6.11.1. Screen dump

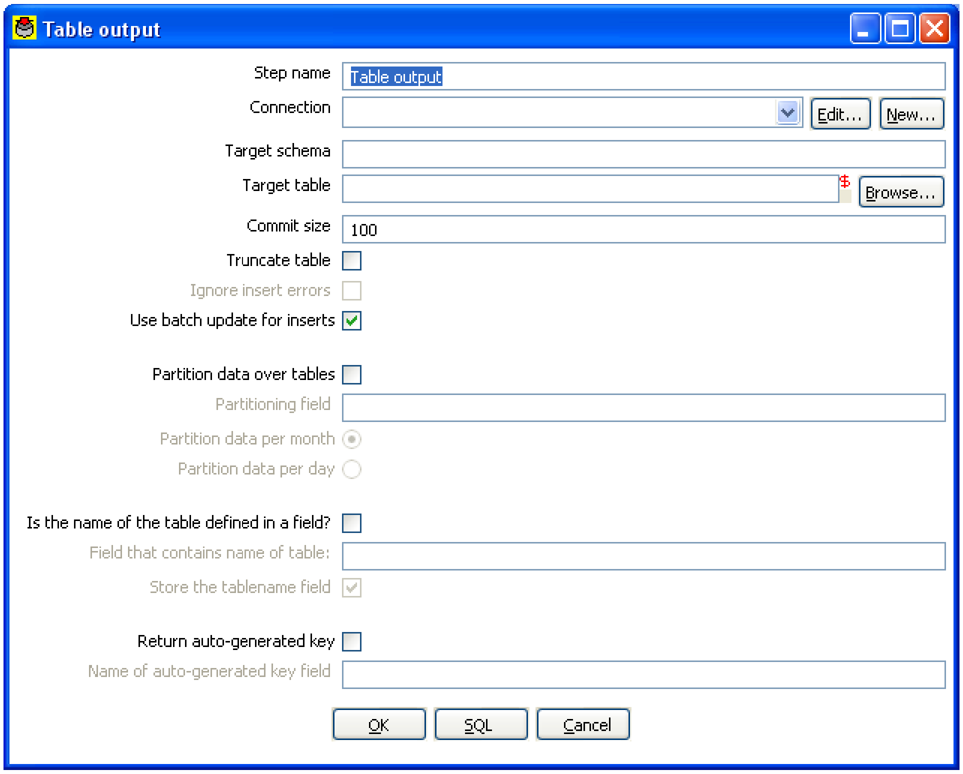


Table output

Step name: Table output

Connection:

Target schema:

Target table:

Commit size: 100

Truncate table □

Ignore insert errors □

Use batch update for inserts ☑

Partition data over tables □

Partitioning field:

Partition data per month ●

Partition data per day ○

Is the name of the table defined in a field? □

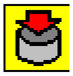
Field that contains name of table:

Store the tablename field ☑

Return auto-generated key □

Name of auto-generated key field:

11.6.11.2. Icon



11.6.11.3. General description

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

11.6.11.4. 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 Schema	The name of the Schema for the table to write data to. This is important for data sources that allow for table names with dots '.' In it.
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) **NOTE:** *Transactions are not supported on all database platforms.*
Truncate table	Select this if you want the table to be truncated before 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	Enable this option if you want to use batch inserts. This feature groups inserts

Option	Description
inserts	statements to limit round trips to the database. This is the fastest option and is enabled by default.

Partition data over tables	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, ...
	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.
	The view SALES allows you to report on the complete sales:
	`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`
	…

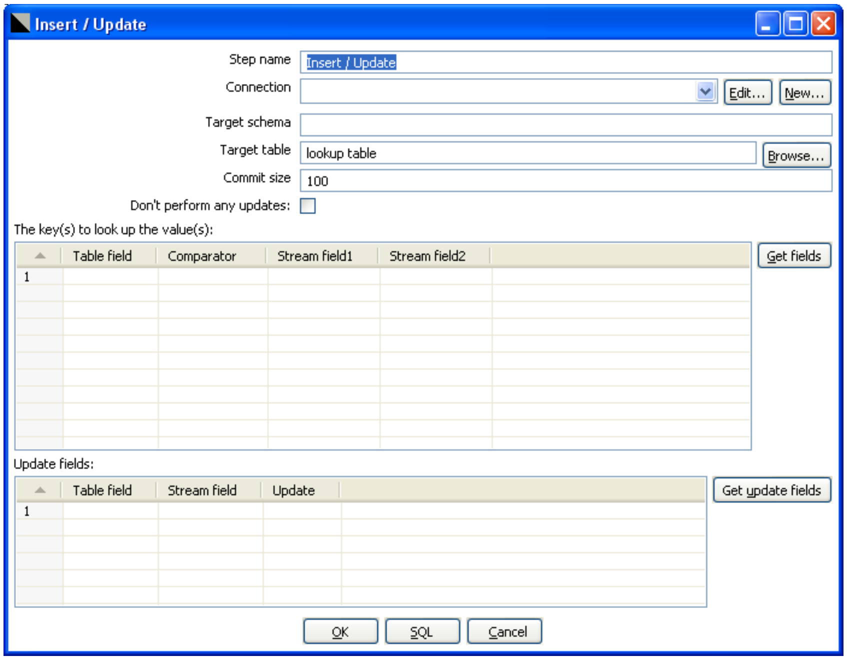
Is the name of the table defined in a field.	Use these options to split the data over one or more tables. The name of the target table is defined in the field you specify. For example if you store customer data in the field gender, the data might end up in tables M and F (Male and Female). There is an option to exclude the field containing the tablename from being inserted into the tables.
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.

11.6.11.5. Extras

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

11.6.12. Insert / Update

11.6.12.1. Screenshot



Insert / Update

Step name: Insert / Update

Connection:

Target schema:

Target table: lookup table

Commit size: 100

Don't perform any updates: □

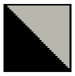
The key(s) to look up the value(s):

	Table field	Comparator	Stream field1	Stream field2
1				

Update fields:

	Table field	Stream field	Update
1			

11.6.12.2. Icon



11.6.12.3. 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.

11.6.12.4. Options

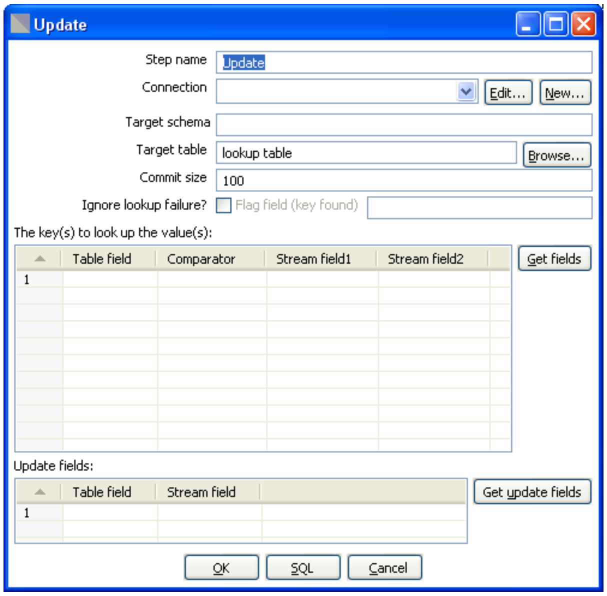
- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **Connection**: Name of the database connection on which the target table resides.
- **Target schema**: The name of the Schema for the table to write data to. This is important for data sources that allow for table names with dots '.' In it.
- **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.

11.6.12.5. 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.

11.6.13. Update

11.6.13.1. Screenshot



Update

Step name: Update

Connection:

Target schema:

Target table: lookup table

Commit size: 100

Ignore lookup failure? □ Flag field (key found)

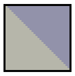
The key(s) to look up the value(s):

	Table field	Comparator	Stream field1	Stream field2
1				

Update fields:

	Table field	Stream field
1		

11.6.13.2. Icon

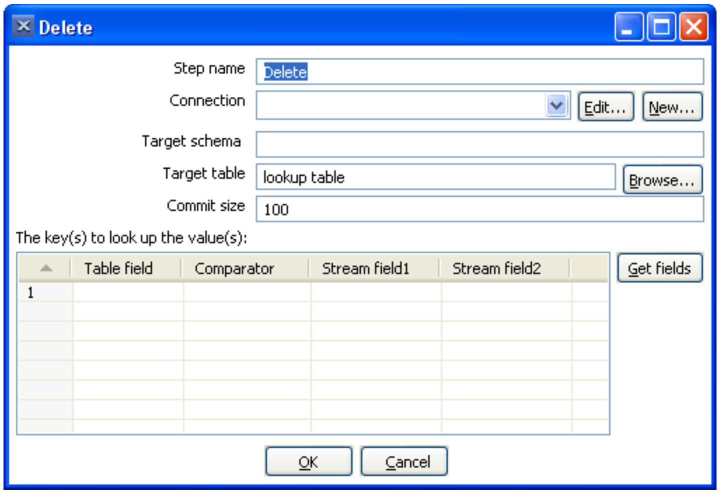


11.6.13.3. General description

This step is the same as the 11.4.13.Insert / Update step, except that no insert is ever done in the database table. ONLY updates are performed.

11.6.14. Delete

11.6.14.1. Screenshot



Delete

Step name: Delete

Connection:

Target schema:

Target table: lookup table

Commit size: 100

The key(s) to look up the value(s):

	Table field	Comparator	Stream field1	Stream field2
1				

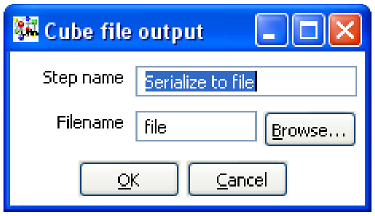
11.6.14.2. Icon

11.6.14.3. General description

This step is the same as the 11.4.15.Delete, except that instead of updating, rows are deleted.

11.6.15. Serialize to file (formerly Cube File Output)

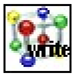
11.6.15.1. Screenshot


Cube file output

Step name: Serialize to file

Filename: file

11.6.15.2. Icon



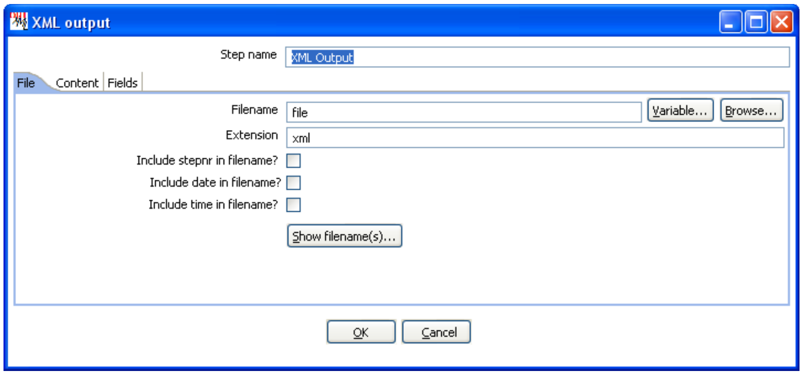
11.6.15.3. General description

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.*

11.6.16. XML Output

11.6.16.1. Screenshots



Step name: XML Output

Filename: file

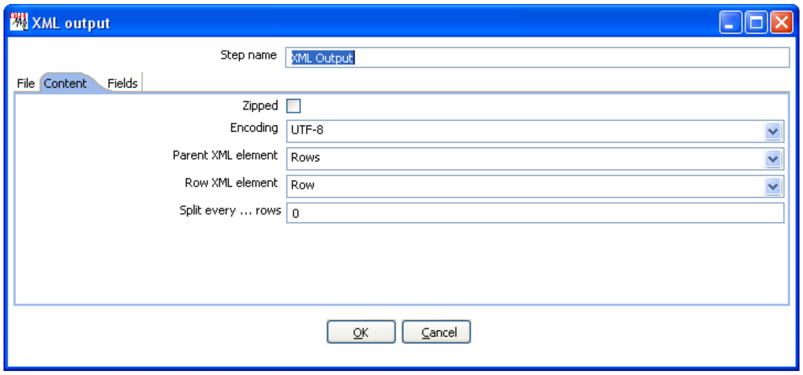
Extension: xml

Include stepnr in filename? ☐

Include date in filename? ☐

Include time in filename? ☐

The "files" tab



Step name: XML Output

Zipped ☐

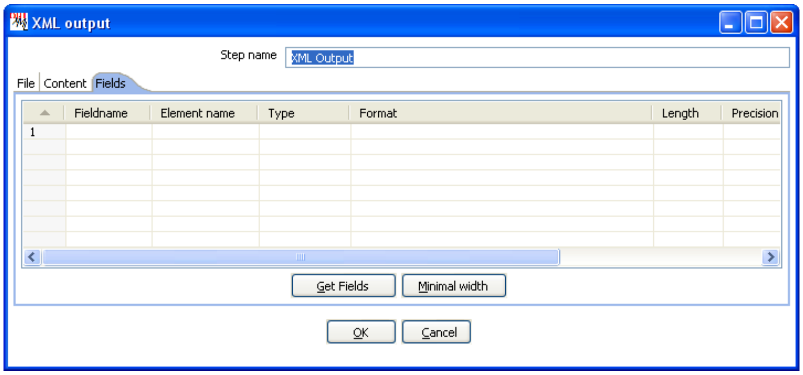
Encoding: UTF-8

Parent XML element: Rows

Row XML element: Row

Split every ... rows: 0

The "Content" tab



Step name: XML Output

	Fieldname	Element name	Type	Format	Length	Precision
1						

The "Fields" tab

11.6.16.2. Icon

11.6.16.3. General description

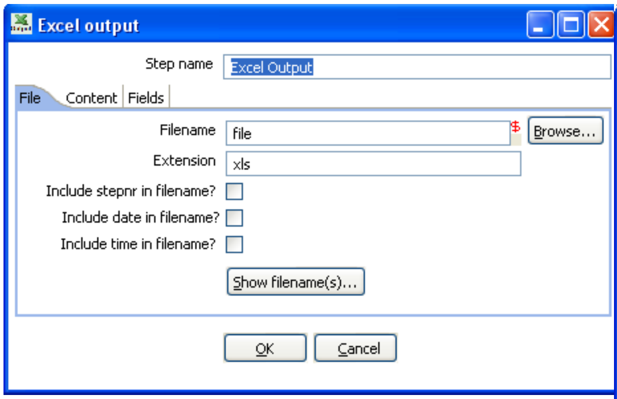
This step allows you to write rows from any source to one or more XML files.

11.6.16.4. 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 10.2. Launching several copies of a step, 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	• Fieldname: Name of the field. • Elementname: the name of the element in the XML file to use. • Type: Type of the field can be either String, Date, or Number. • Format mask to convert with: see 8.5.1.5.1. Number formats for a complete description of format specifiers. • Length: Output string is padded to this length if it is specified. • Precision: The precision to use. • Currency: Symbol used to represent currencies 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 "," (10,000.00) or "." (5.000,00) • Null: the string to use in case the field value is null.

11.6.17. Excel Output

11.6.17.1. Screenshots



Excel output

Step name: Excel Output

Filename: file

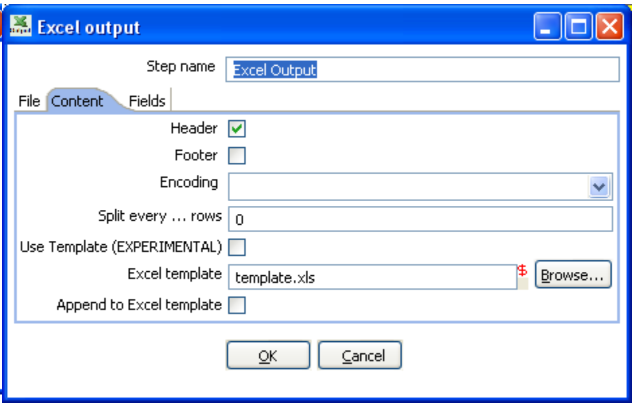
Extension: xls

Include stepnr in filename? □

Include date in filename? □

Include time in filename? □

The "File" tab



Excel output

Step name: Excel Output

Header ☑

Footer □

Encoding:

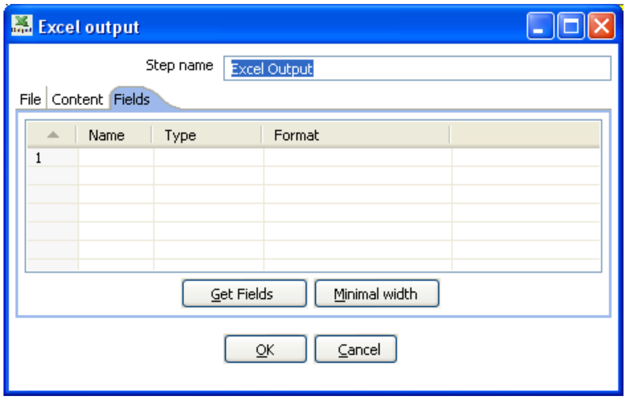
Split every ... rows: 0

Use Template (EXPERIMENTAL) □

Excel template: template.xls

Append to Excel template □

The "Content" tab



Excel output

Step name: Excel Output

	Name	Type	Format
1			

The "Fields" tab

11.6.17.2. Icon

11.6.17.3. General description

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

11.6.17.4. Options

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

File tab:

- The filename and extension, with variable support
- The extension to use on the output file
- An option to include the stepnr in the filename (allowing parallel file creation)

- An option to include the date and/or the time in the filename
- A button to show what the filenames would look like after running.

Content tab:

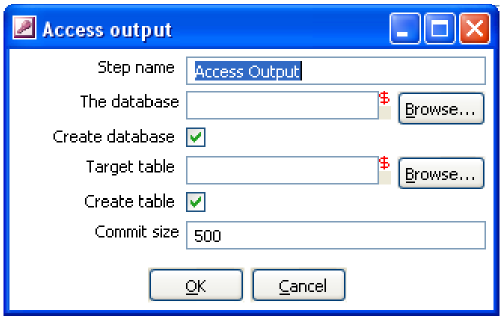
- Header: Check if the spreadsheet needs a header above the exported rows of data.
- Footer: Check if the spreadsheet needs a footer below the exported rows of data.
- Encoding: Specify the encoding of the spreadsheet, leave empty to keep the default for the platform.
- Split ever ... rows: splits the data over several output files. (each in it's own spreadsheet)
- Use Template: this is an experimental feature that requires testing. Check this to use a template when outputting data to Excel.
- Excel template: the name of the template used to format the Excel output file
- Append to Excel template: Check this option to have the output appended to the Excel template specified

Fields tab:

- Here you can specify the fields that need to be exported to the Excel file(s). A button "Get fields is provided to automatically fill in the available fields if the sheets have a header row.
- You can specify the format as used in Excel (NOT Kettle)

11.6.18. Microsoft Access Output

11.6.18.1. Screenshot



Access output

Step name: Access Output

The database:

Create database ☑

Target table:

Create table ☑

Commit size: 500

11.6.18.2. Icon

11.6.18.3. General Description

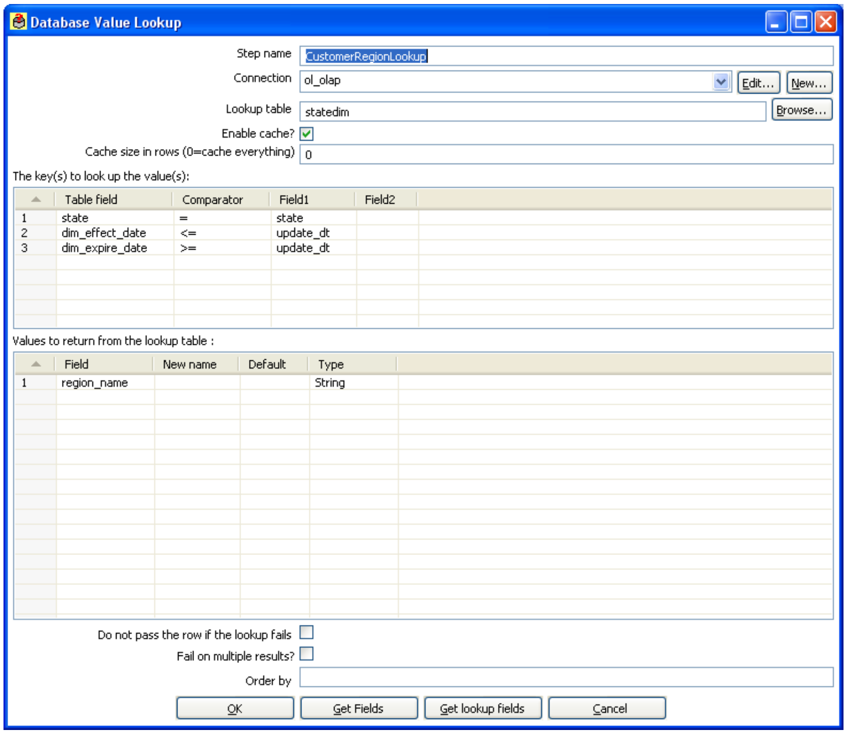
This allows you to create a new Access database file as an output in a transformation.

11.6.18.4. Options

- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **The database filename**: The filename of the database file you are connecting to.
- **Create database**: Check this to generate a new Access database file.
- **Target table**: Specify the table you want to output data to.
- **Create table**: Check this to create a new table in the Access database.
- **Commit size**: Defines the commit size when outputting data.

11.6.19. Database lookup

11.6.19.1. Screen dump



Step name: CustomerRegionLookup

Connection: ol_olap

Lookup table: statedim

Enable cache? ☑

Cache size in rows (0=cache everything): 0

The key(s) to look up the value(s):

	Table field	Comparator	Field1	Field2
1	state	=	state	
2	dim_effect_date	<=	update_dt	
3	dim_expire_date	>=	update_dt	

Values to return from the lookup table :

	Field	New name	Default	Type
1	region_name			String

Do not pass the row if the lookup fails ☐

Fail on multiple results? ☐

Order by:

11.6.19.2. Icon

11.6.19.3. General description

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

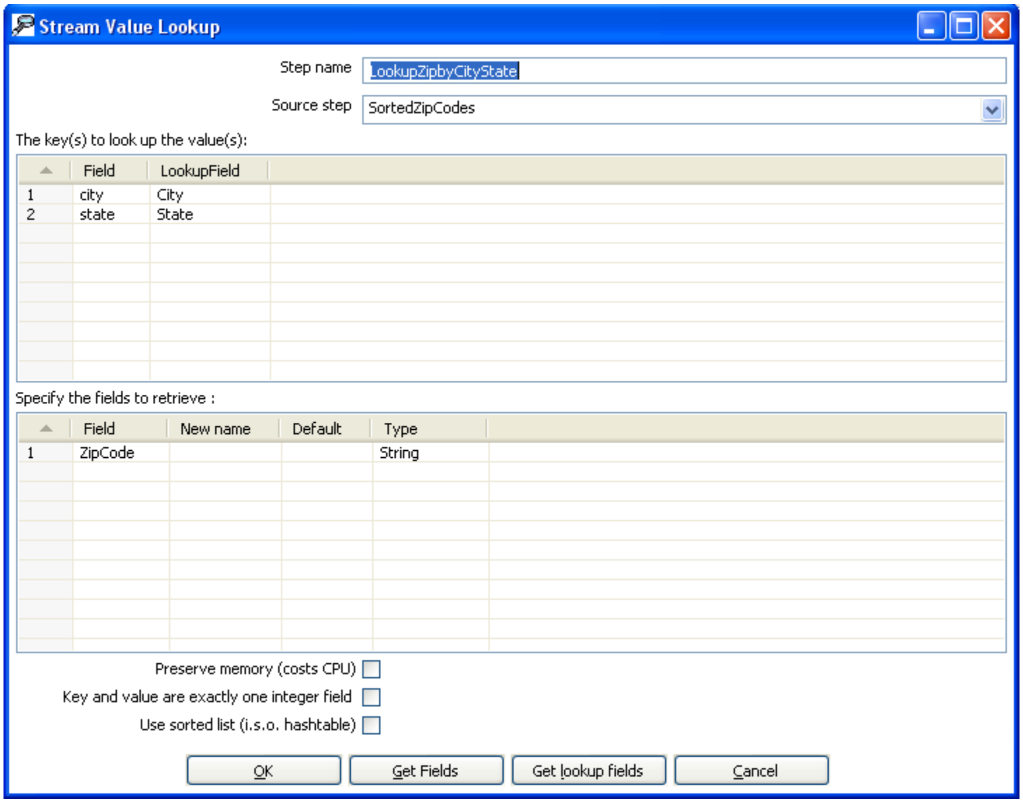
11.6.19.4. Options

- **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 11.2.Launching several copies of a step

11.6.20. Stream lookup

11.6.20.1. Screenshot



Step name: LookupZipbyCityState

Source step: SortedZipCodes

The key(s) to look up the value(s):

	Field	LookupField
1	city	City
2	state	State

Specify the fields to retrieve :

	Field	New name	Default	Type
1	ZipCode			String

Preserve memory (costs CPU) □

Key and value are exactly one integer field □

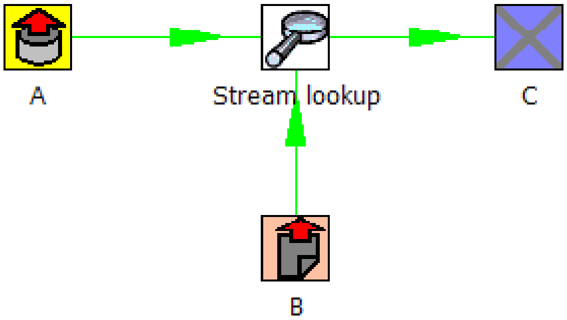
Use sorted list (i.s.o. hashtable) □

11.6.20.2. Icon

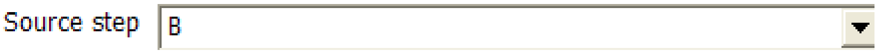
11.6.20.3. General description

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: B

11.6.20.4. Options

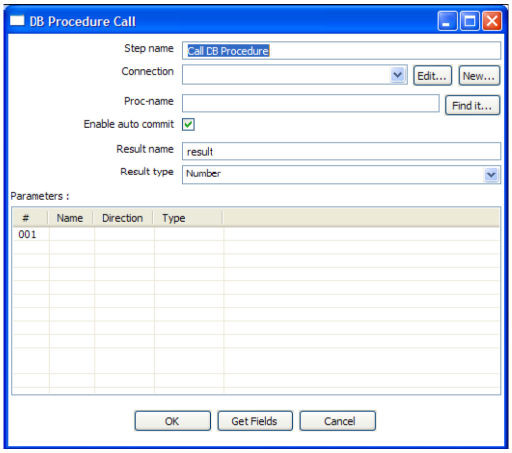
- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **Source step**: This is the step name where the lookup data is coming from
- **The key(s) to lookup value(s)**: Allows you to specify the names of the fields that are used to lookup values. Values are always 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.
- **Preserve Memory**: This will encode rows of data to preserve memory while sorting.
- **Key and value are exactly one integer field**: This will also preserve memory while executing a sort.
- **Use sorted list**: Check this to store values using a sorted list. This provides better memory usage when working with data sets containing wide rows.

11.6.20.5. 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.

11.6.21. Call DB Procedure

11.6.21.1. Screenshot



Step name: Call DB Procedure

Connection:

Proc-name:

Enable auto commit ☑

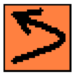
Result name: result

Result type: Number

Parameters :

#	Name	Direction	Type
001			

11.6.21.2. Icon



11.6.21.3. General description

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

11.6.21.4. 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.

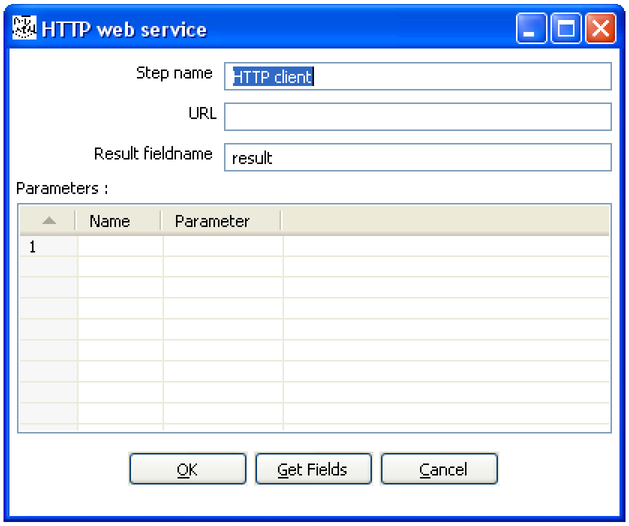
11.6.21.5. Extra

- "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

11.6.22. HTTP Client

11.6.22.1. Screen shot



Step name: HTTP client

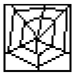
URL:

Result fieldname: result

Parameters :

	Name	Parameter
1		

11.6.22.2. Icon



11.6.22.3. General Description

The HTTP client step performs a very simple call to a base URL with options appended to it like this:

```
http://<URL>?param1=value1&param2=value2&..
```

The result is stored in a String field with the specified name.

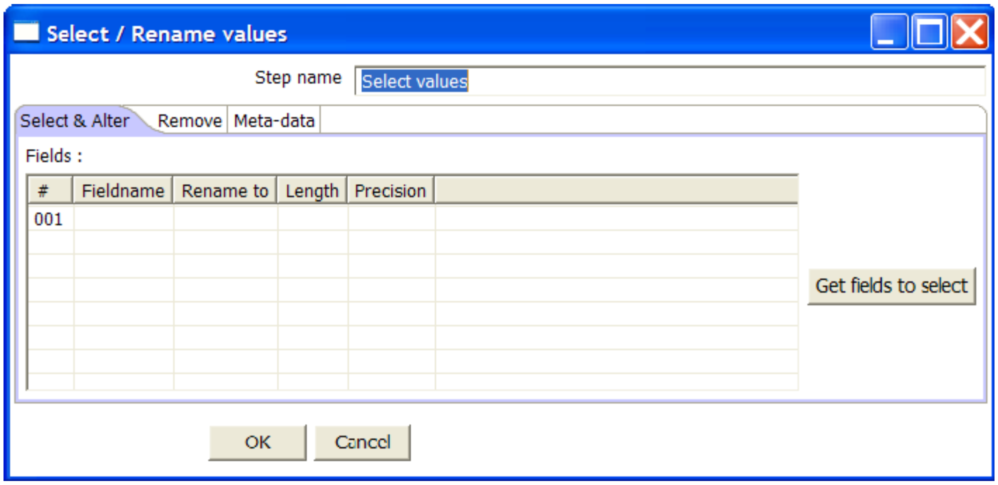
11.6.22.4. Options

- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **URL**: the base URL string
- **Result fieldname**: the name of the field to store results
- **Parameters**: this section is where you define the parameter name-value pairs to pass on the URL

11.6.23. Select values

11.6.23.1. Screen dump

The "Select & Alter" tab:

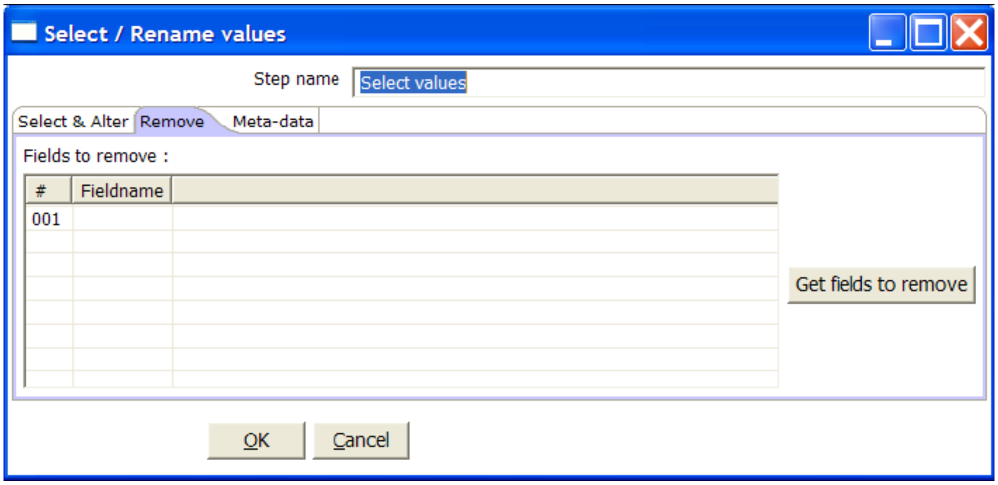


Step name: Select values

Fields :

#	Fieldname	Rename to	Length	Precision
001				

The "Remove" tab:

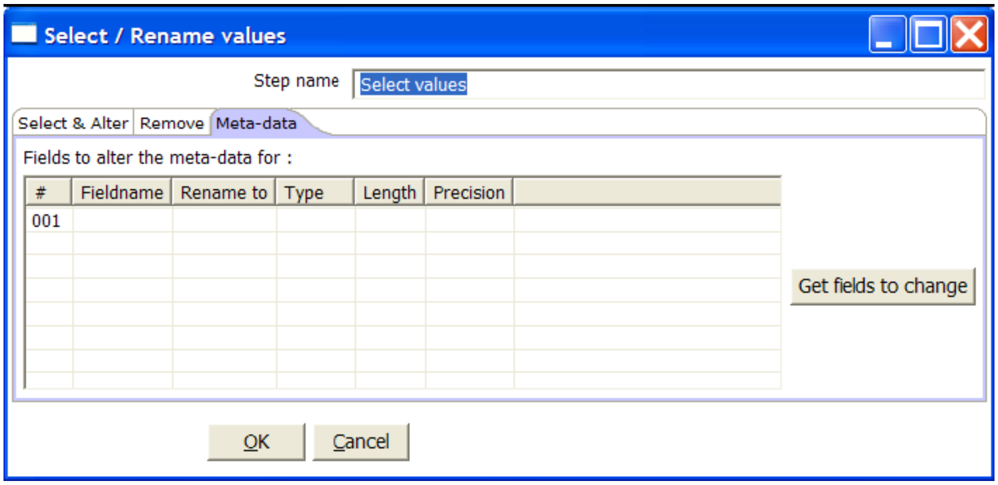


Step name: Select values

Fields to remove :

#	Fieldname
001	

The "Meta-data" tab:

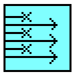


Step name: Select values

Fields to alter the meta-data for :

#	Fieldname	Rename to	Type	Length	Precision
001					

11.6.23.2. Icon



11.6.23.3. 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:

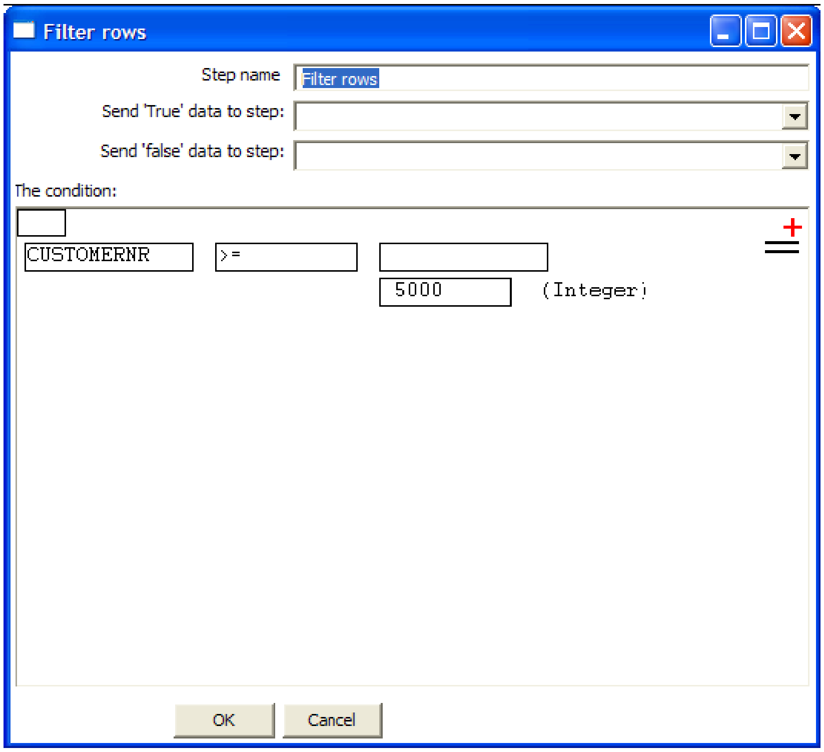
- Select & Alter: Specify the exact order and name in which the fields have to be placed in the output rows.
- Remove: Specify the fields that have to be removed from the output rows.
- Meta-data: Change the name, type, length and precision (the meta-data) of one or more fields.

11.6.23.4. 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)

11.6.24. Filter rows

11.6.24.1. Screenshot



Step name: Filter rows

Send 'True' data to step:

Send 'false' data to step:

The condition:

CUSTOMERNR >= 5000 (Integer)

11.6.24.2. Icon

11.6.24.3. General description

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

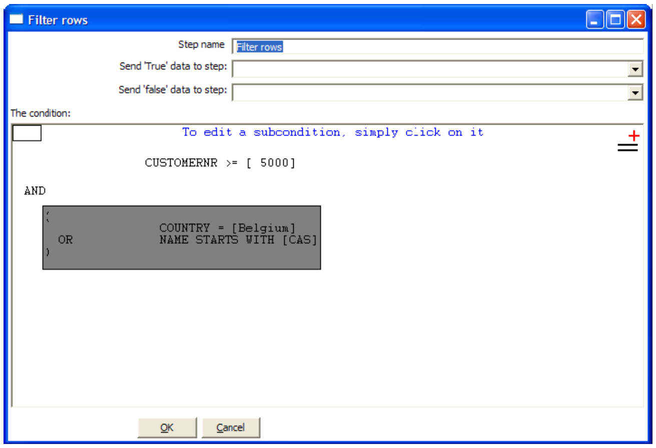
Once this step is connected to a previous step (one or more and receiving input), you can simply click on the "<field>", "=" and "<value>" areas to construct a condition.

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:



Step name: Filter rows

Send 'True' data to step:

Send 'false' data to step:

The condition:

To edit a subcondition, simply click on it

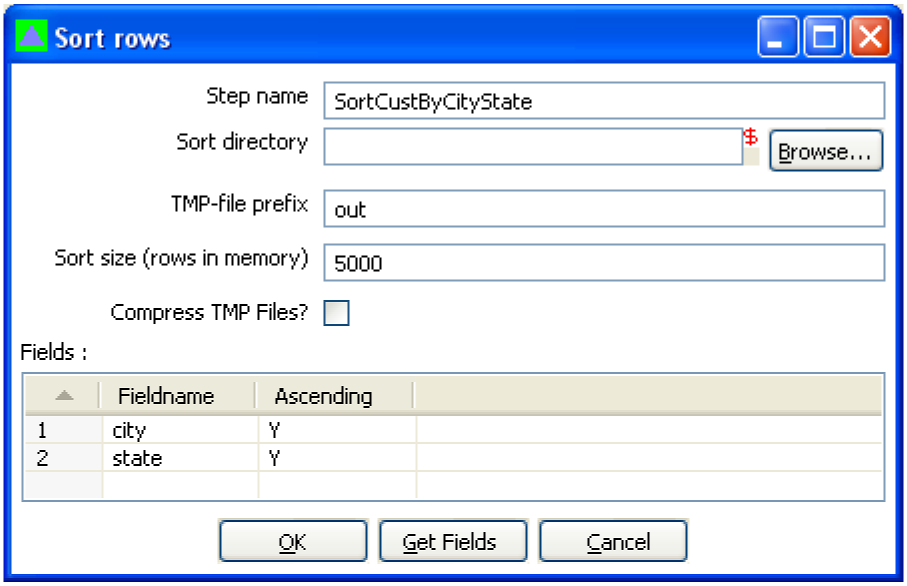
```
CUSTOMERNR >= [ 5000]

AND
(
 OR           COUNTRY = [Belgium]
              NAME STARTS WITH [CAS]
)
```

11.6.24.4. 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.

11.6.25. Sort rows



Sort rows

Step name: SortCustByCityState

Sort directory:

TMP-file prefix: out

Sort size (rows in memory): 5000

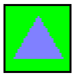
Compress TMP Files? □

Fields :

	Fieldname	Ascending
1	city	Y
2	state	Y

11.6.25.1. Screenshot

11.6.25.2. Icon



11.6.25.3. 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.*

11.6.25.4. 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.

TMP-file prefix: Choose a recognizable prefix in order to recognize the files when they show up in the temp directory.

Sort size (rows in memory): 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.

Compress TMP Files?: This option compresses temporary files when they are needed to complete the sort.

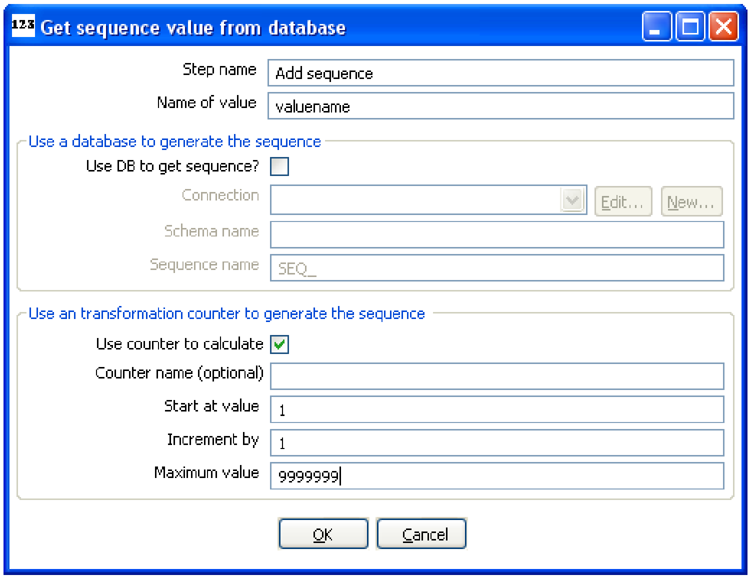
A list of fields and whether they should be sorted ascending or not.

11.6.25.5. Extras

The "Get fields" button inserts the fields into the "Fields" grid if the step is connected to theprevious ones.

11.6.26. Add sequence

11.6.26.1. Screen dump



Get sequence value from database

Step name: Add sequence

Name of value: valuename

Use a database to generate the sequence

Use DB to get sequence? □

Connection:

Schema name:

Sequence name: SEQ_

Use an transformation counter to generate the sequence

Use counter to calculate ☑

Counter name (optional):

Start at value: 1

Increment by: 1

Maximum value: 9999999

11.6.26.2. Icon

11.6.26.3. 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.*

11.6.26.4. 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.
- **Counter name (optional)**: if multiple steps in a transformation generate the same value name, this option would allow you to specify the name of the counter you want to associate the sequence with. This would avoid forcing unique sequencing across multiple steps.

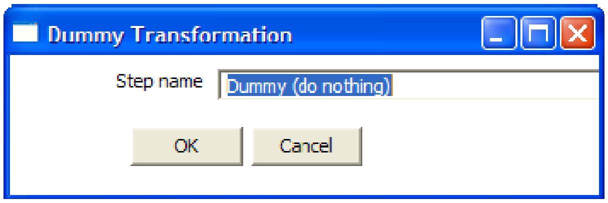
- **Start at**: give the start value of the sequence.
- **Increment by**: give the increment of the sequence.
- **Maximum value**: this is the maximum value after which the sequence will start back at the start value (Start At).

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, …
```

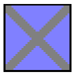
11.6.27. Dummy (do nothing)

11.6.27.1. Screenshot



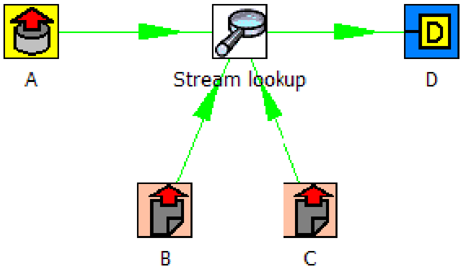
Step name: Dummy (do nothing)

11.6.27.2. Icon

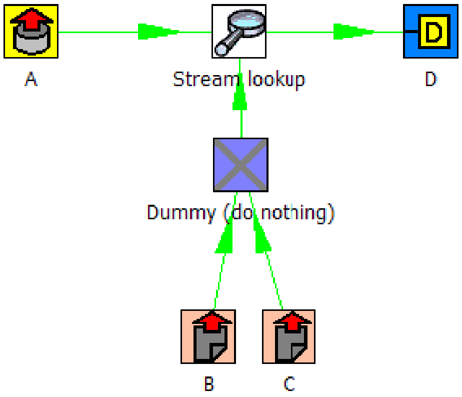


11.6.27.3. General description

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:

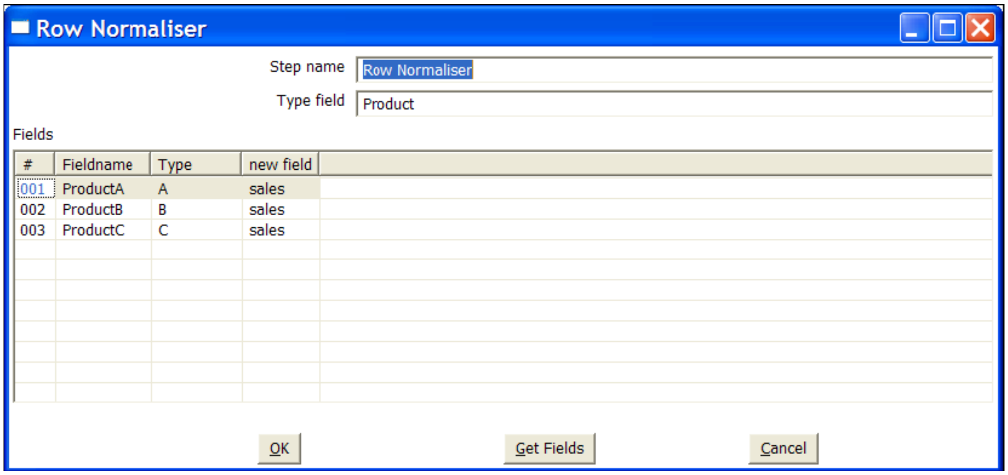


11.6.27.4. Options

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

11.6.28. Row Normaliser

11.6.28.1. Screenshot



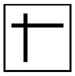
Step name: Row Normaliser

Type field: Product

Fields

#	Fieldname	Type	new field
001	ProductA	A	sales
002	ProductB	B	sales
003	ProductC	C	sales

11.6.28.2. Icon



11.6.28.3. 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.

11.6.28.4. 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).

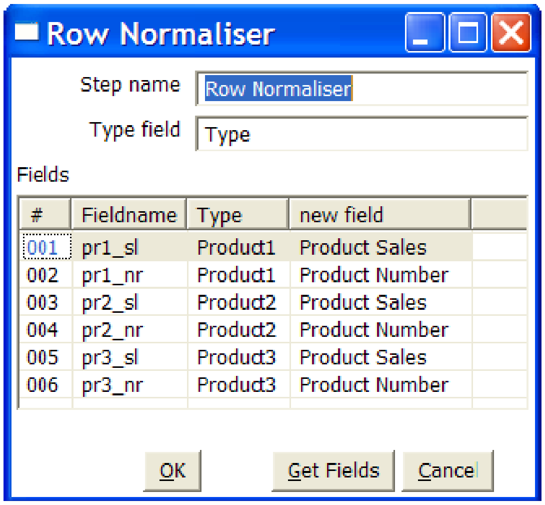
NOTE: *It is possible to normalize more then on field at a time.* Consider this example:

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:



Step name: Row Normaliser

Type field: Type

Fields

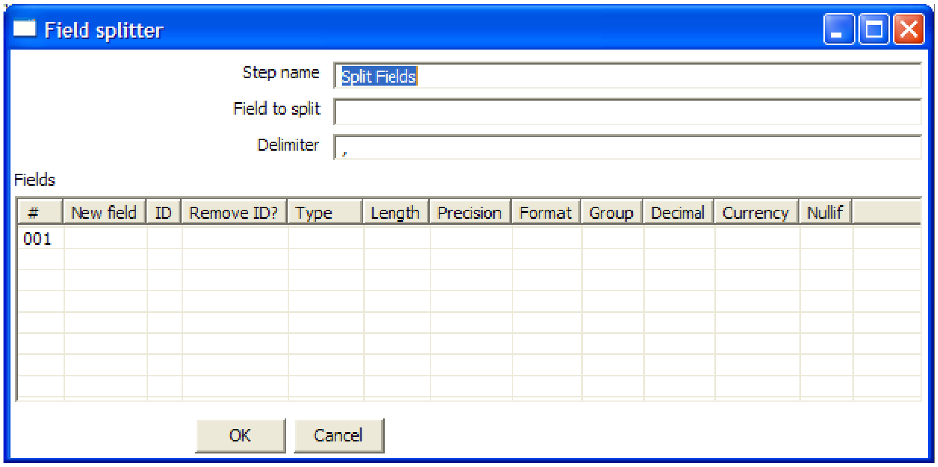
#	Fieldname	Type	new field
001	pr1_sl	Product1	Product Sales
002	pr1_nr	Product1	Product Number
003	pr2_sl	Product2	Product Sales
004	pr2_nr	Product2	Product Number
005	pr3_sl	Product3	Product Sales
006	pr3_nr	Product3	Product Number

11.6.28.5. Extra

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

11.6.29. Split Fields

11.6.29.1. Screenshot



Step name: Split Fields

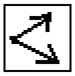
Field to split:

Delimiter: ,

Fields

#	New field	ID	Remove ID?	Type	Length	Precision	Format	Group	Decimal	Currency	Nullif
001											

11.6.29.2. Icon



11.6.29.3. General description

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

11.6.29.4. 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

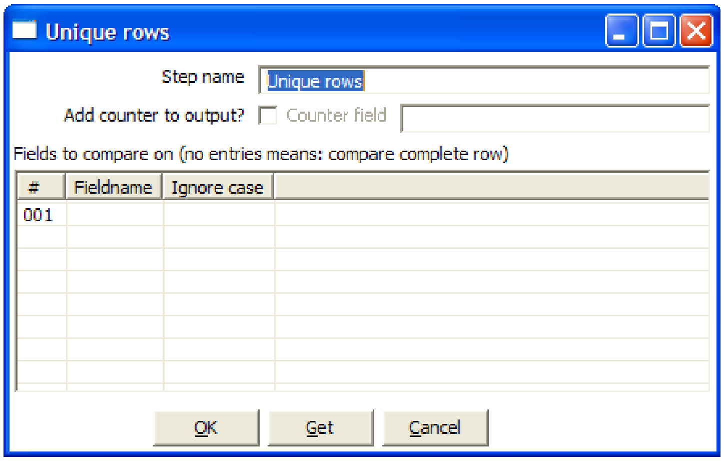
Example: SALES_VALUES field containing "Sales2=310.50, Sales4=150.23"

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

11.6.30. Unique rows

11.6.30.1. Screenshot



Step name: Unique rows

Add counter to output? □ Counter field

Fields to compare on (no entries means: compare complete row)

#	Fieldname	Ignore case
001		

11.6.30.2. Icon

11.6.30.3. 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 :-))

11.6.30.4. Options

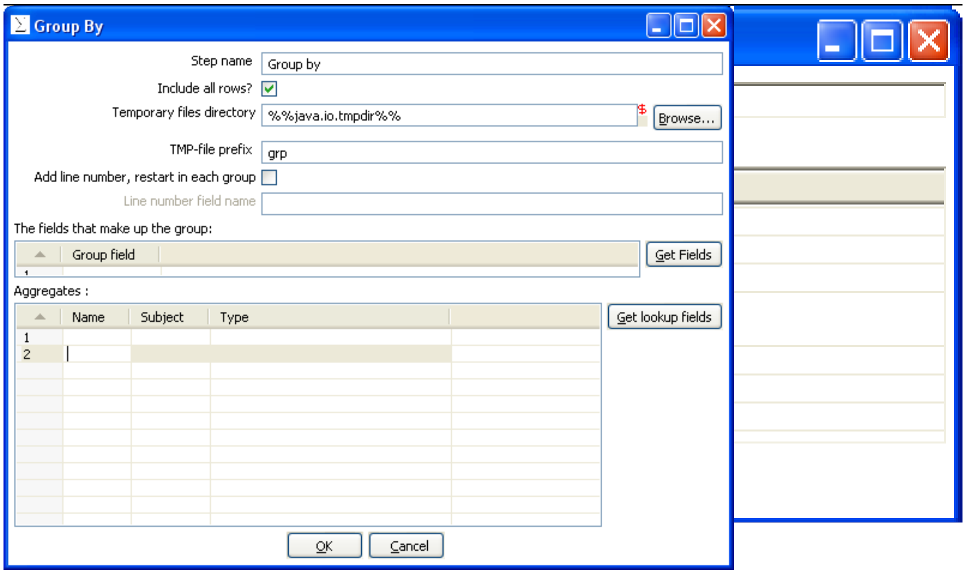
- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **Add counter to output?**: Enable this if you want to know how many rows were doubles for each row in the output.
- **Counter fields**: Name of the numeric field containing the number of rows
- **Fieldnames**: 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).

11.6.30.5. Extra

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

11.6.31. Group By

11.6.31.1. Screenshot



Group By

Step name: Group by

Include all rows? ☑

Temporary files directory: %%java.io.tmpdir%%

TMP-file prefix: grp

Add line number, restart in each group ☐

Line number field name:

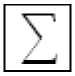
The fields that make up the group:

Group field

Aggregates :

	Name	Subject	Type
1			
2			

11.6.31.2. Icon



11.6.31.3. General description

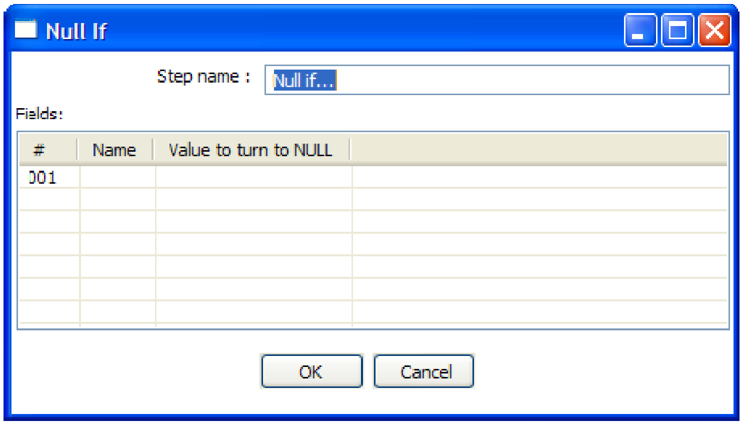
This step allows you to calculate values over a defined group of fields.
Examples that are common are: calculate the average sales per product, get the number of yellow shirts that we have in stock, etc.

11.6.31.4. 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)
- **Temporary files 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.
- **TMP-file prefix**: Specify the file prefix used when naming temporary files.
- **Add line number, restart in each group**: check this if you want to add a line number that restarts at 1 in each group.
- **Line number field name**: Specify the name of the field in which the line numbers will be inserted.

11.6.32. Null If

11.6.32.1. Screenshot

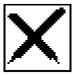


Step name : Null if...

Fields:

#	Name	Value to turn to NULL
)01		

11.6.32.2. Icon

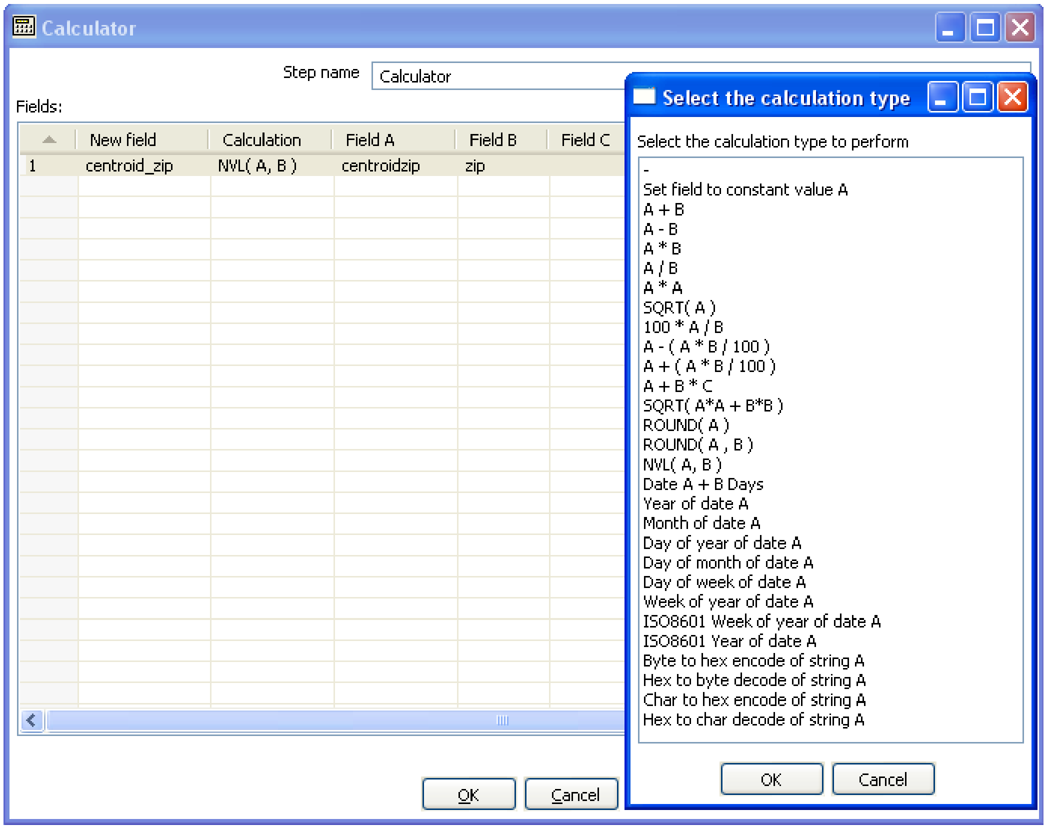


11.6.32.3. General description

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

11.6.33. Calculator

11.6.33.1. Screenshot



Step name: Calculator

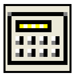
Fields:

	New field	Calculation	Field A	Field B	Field C
1	centroid_zip	NVL(A, B)	centroidzip	zip	

Select the calculation type to perform

-
Set field to constant value A
A + B
A - B
A * B
A / B
A * A
SQRT(A)
100 * A / B
A - (A * B / 100)
A + (A * B / 100)
A + B * C
SQRT(A*A + B*B)
ROUND(A)
ROUND(A , B)
NVL(A, B)
Date A + B Days
Year of date A
Month of date A
Day of year of date A
Day of month of date A
Day of week of date A
Week of year of date A
ISO8601 Week of year of date A
ISO8601 Year of date A
Byte to hex encode of string A
Hex to byte decode of string A
Char to hex encode of string A
Hex to char decode of string A

11.6.33.2. Icon



11.6.33.3. 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, simple write an e-mail to 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.

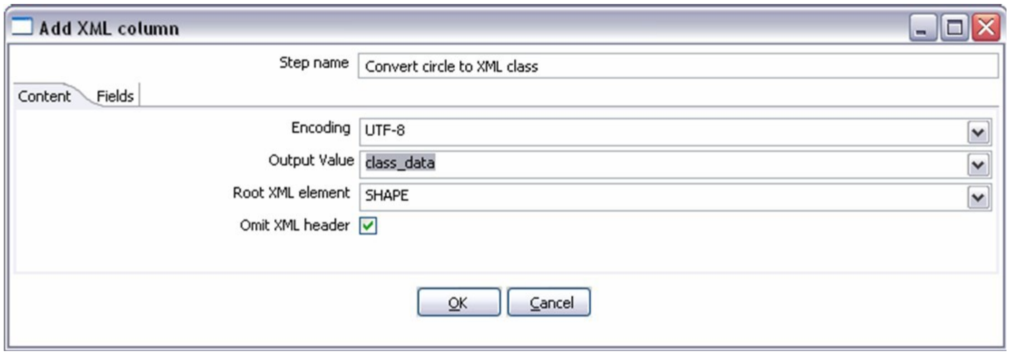
Besides the arguments (Field A, Field B and Field C) you also need to specify the return type of the function. You can also opt to remove the field from the result (output) after all values were calculated. This is useful for removing temporary values.

11.6.33.4. 10.4.34.4 Function List

Function	Description	Required fields
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 √(A2+B2)	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
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
Byte to hex encode of string A	Encode bytes in a string to a hexadecimal representation	A
Hex encode of string A	Encode a string in its own hexadecimal representation	A
Char to hex encode of string A	Encode characters in a string to a 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

11.6.34. XML Add

11.6.34.1. Screenshots



Step name: Convert circle to XML class

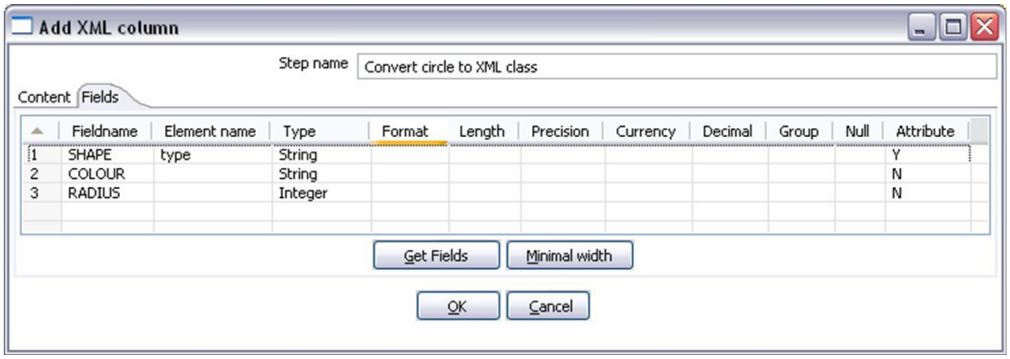
Encoding: UTF-8

Output Value: class_data

Root XML element: SHAPE

Omit XML header: ☑

The "Content" tab



Step name: Convert circle to XML class

#	Fieldname	Element name	Type	Format	Length	Precision	Currency	Decimal	Group	Null	Attribute
1	SHAPE	type	String								Y
2	COLOUR		String								N
3	RADIUS		Integer								N

The "fields" tab

11.6.34.2. Icon

11.6.34.3. General description

This step allows you to encode the content of a number of fields in a row in XML. This XML is added to the row in the form of a String field.

11.6.34.4. 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 10.2. Launching several copies of a step, 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

Option	Description
	is created.
Encoding	The encoding to use. This encoding is specified in the header of the XML file.
Output Value	The name of the new field that will contain the XML
Root XML element	The name of the root element in the generated XML.
Fields	• Fieldname: Name of the field. • Element name: the name of the element in the XML file to use. • Type: Type of the field can be either String, Date, or Number. • Format mask to convert with: see 10.4.1.5.1. Number formats for a complete description of format specifiers. • Length: Output string is padded to this length if it is specified. • Precision: The precision to use. • Currency: Symbol used to represent currencies 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 "," (10,000.00) or "." (5.000,00) • Null: the string to use in case the field value is null. • Attribute: make this an attribute (N means : element)

11.6.34.5. Example

11.6.34.5.1. Motivation

I have data that comes in a variety of classes and I would like to store it as XML in my database i.e. I want to turn the raw data into the database layout below.

11.6.34.5.2. Raw data

SHAPE	COLOUR	id	X	Y	RADIUS
circle	blue	1	3	5	5
circle	red	2	1	3	5
circle	blue	5	5	9	5
circle	blue	6	8	2	5
circle	red	7	9	7	5

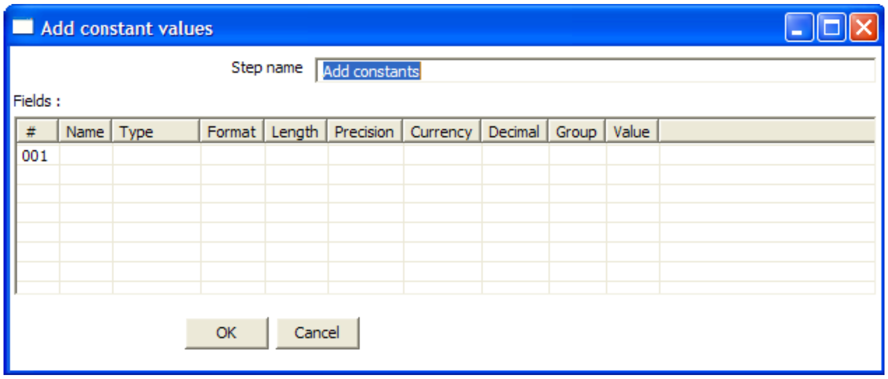
SHAPE	COLOUR	id	X	Y	LENGTH	WIDTH
rectangle	blue	3	3	1	6	4
rectangle	red	6	2	4	6	4
rectangle	blue	10	8	2	6	4
rectangle	red	12	7	8	6	4
rectangle	Blue	14	5	2	6	4

11.6.34.5.3. Output sample

ID	X	Y	CLASS_DATA
3	4	7	<SHAPE type="circle"><COLOUR>blue</COLOUR><RADIUS> 5</RADIUS></SHAPE>
1	6	3	<SHAPE type="rectangle"><COLOUR>blue</COLOUR><WIDTH> 4</WIDTH><LENGTH> 6</LENGTH></SHAPE>
2	8	8	<SHAPE type="rectangle"><COLOUR>blue</COLOUR><WIDTH> 4</WIDTH><LENGTH>6</LENGTH></SHAPE>
5	5	2	<SHAPE type="circle"><COLOUR>blue</COLOUR><RADIUS> 5</RADIUS></SHAPE>

11.6.35. Add constants

11.6.35.1. Screenshot



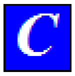
Add constant values

Step name: Add constants

Fields :

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value
001									

11.6.35.2. Icon



11.6.35.3. General description and usage

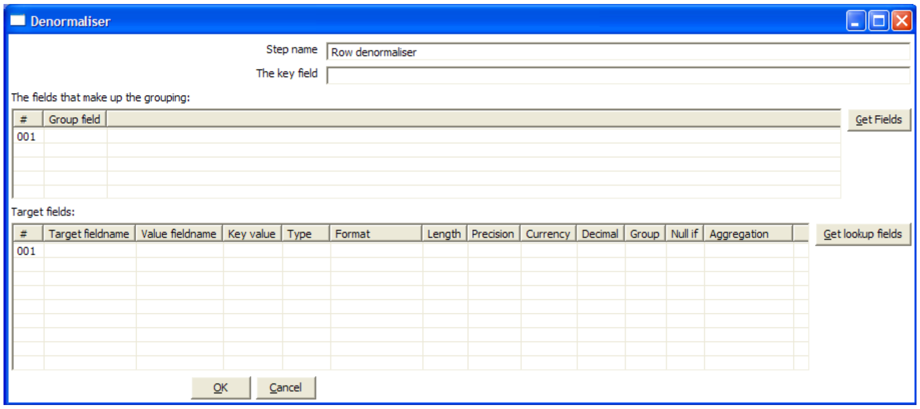
This step is the easiest and most performing way to add constants to a stream.

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.

11.6.36. Row Denormaliser

11.6.36.1. Screenshot



Step name: Row denormaliser

The key field:

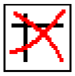
The fields that make up the grouping:

#	Group field
001	

Target fields:

#	Target fieldname	Value fieldname	Key value	Type	Format	Length	Precision	Currency	Decimal	Group	Null if	Aggregation
001												

11.6.36.2. Icon



11.6.36.3. General description

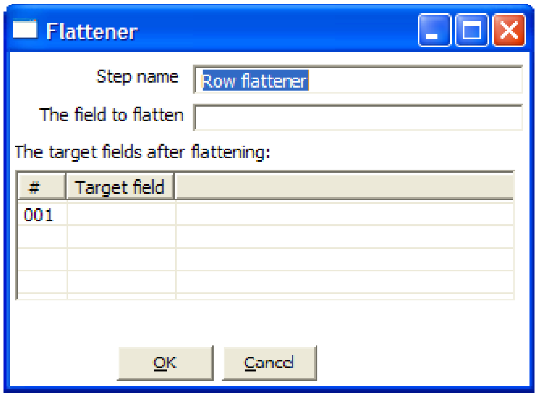
This step allows you de-normalise data by looking up key-value pairs.
It also allows you to immediately convert data types.

11.6.36.4. 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	Specify the fields to de-normalise. You do it by specifying the String value for the key field (see above). 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. In case you get key-value pair collisions (key is not unique for the group specified) you can specify the aggregation method to use.

11.6.37. Flattener

11.6.37.1. Screenshot



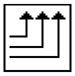
Step name: Row flattener

The field to flatten:

The target fields after flattening:

#	Target field
001	

11.6.37.2. Icon



11.6.37.3. General description

This step allows you flatten sequentially provided data.

11.6.37.4. Options

Option	Description
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.

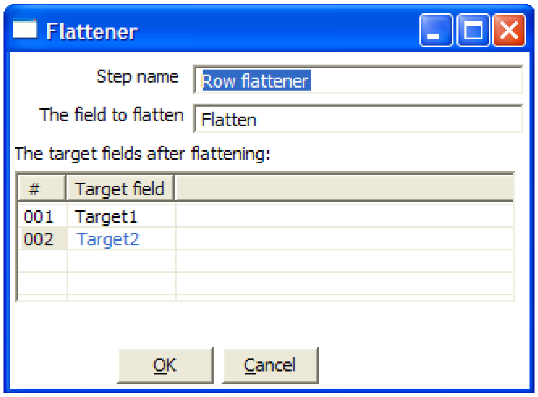
11.6.37.5. Example

Field1	Field2	Field3	Flatten
A	B	C	One
A	B	C	Two
D	E	F	Three
D	E	F	Four

Can be flattened to:

Field1	Field2	Field3	Target1	Target2
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

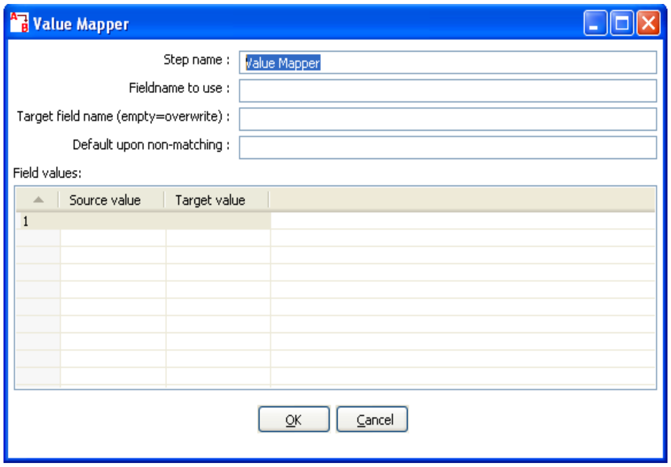
The field to flatten: Flatten

The target fields after flattening:

#	Target field
001	Target1
002	Target2

11.6.38. Value Mapper

11.6.38.1. Screenshot



Value Mapper

Step name : Value Mapper

Fieldname to use :

Target field name (empty=overwrite) :

Default upon non-matching :

Field values:

	Source value	Target value
1		

11.6.38.2. Icon

11.6.38.3. General description

This step 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

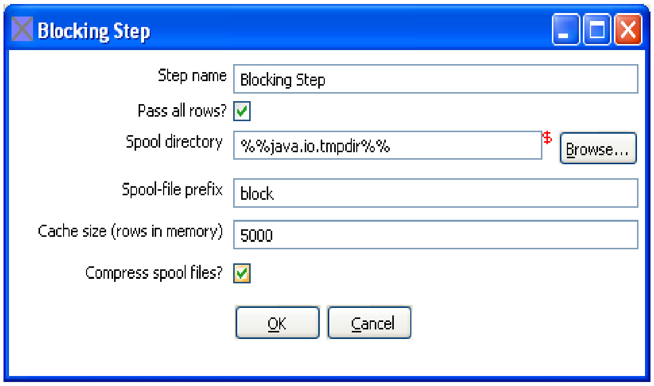
Source/Target: EN/English, FR/French, NL/Dutch, ES/Spanish, DE/German, ...

NOTE: It is also possible to convert a null field or empty String value to a non-empty value. Leave the “Source value” field empty for this. It is obviously only possible to specify one of these empty source field values.

NOTE: If the source value is not empty, but there's no match you can specify a default via “Default upon non-matching”.

11.6.39. Blocking step

11.6.39.1. Screenshot



Blocking Step

Step name: Blocking Step

Pass all rows? ☑

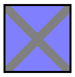
Spool directory: %%java.io.tmpdir%%

Spool-file prefix: block

Cache size (rows in memory): 5000

Compress spool files? ☑

11.6.39.2. Icon



11.6.39.3. General description

This is a very simple step. It blocks all output until the very last row was received from the previous step. At that time either the last row is then sent off to the next step or the complete input is sent of to the next step. This step then knows that all previous steps have finished.

You can use this for triggering plugins, stored procedures, java scripts, ... or for synchronization purposes.

11.6.39.4. Options

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

Pass all rows?: Determines whether to pass 1 rows or all rows.

Spool 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.

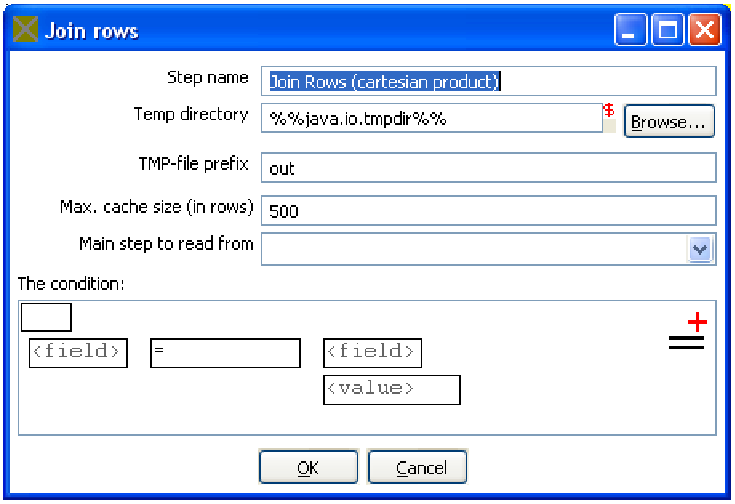
Spool-file prefix: Choose a recognizable prefix in order to recognize the files when they show up in the temp directory.

Cache size (rows in memory): The more rows you can store in memory, the faster the step works.

Compress spool files?: This option compresses temporary files when they are needed.

11.6.40. Join Rows (Cartesian product)

11.6.40.1. Screenshot



Join rows

Step name: Join Rows (cartesian product)

Temp directory: %%java.io.tmpdir%%

TMP-file prefix: out

Max. cache size (in rows): 500

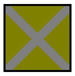
Main step to read from:

The condition:

<field> = <field>

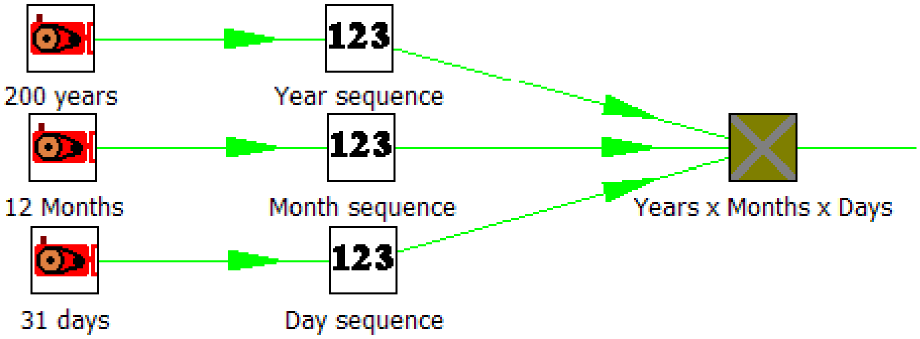
<value>

11.6.40.2. Icon



11.6.40.3. General description

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



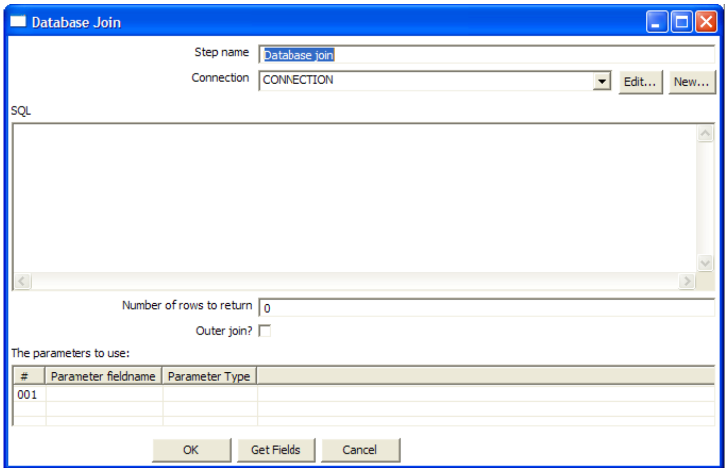
The "Years x Months x Days" step outputs all combinations of Year, Month and Day. (1900, 1, 1 → 2100, 12, 31) and can be used to create a date dimension.

11.6.40.4. 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.

11.6.41. Database Join

11.6.41.1. Screenshot



Step name: Database join

Connection: CONNECTION

SQL

Number of rows to return: 0

Outer join? ☐

The parameters to use:

#	Parameter fieldname	Parameter Type
001		

11.6.41.2. Icon

11.6.41.3. 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
```

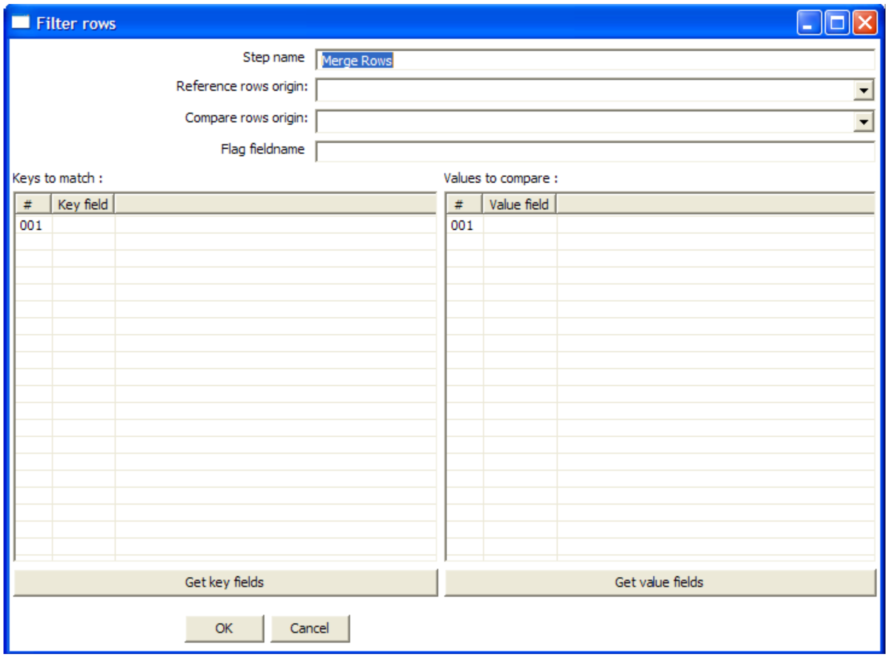
All you then need to specify as a parameter is the productnr and you'll get the customernr included in the result.

11.6.41.4. Options

- **Step name**: Name of the step. This name has to be unique in a single transformation.
- The database connection to use.
- **SQL**: 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.

11.6.42. Merge rows

11.6.42.1. Screenshot



Filter rows

Step name: Merge Rows

Reference rows origin:

Compare rows origin:

Flag fieldname

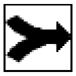
Keys to match :

#	Key field
001	

Values to compare :

#	Value field
001	

11.6.42.2. Icon



11.6.42.3. 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 date of last update.
The 2 streams of rows, a reference stream (the old data) and a compare stream (the new data), are merged. Each time only the last version of a row is passed onto the next steps. The row is marked as follows:

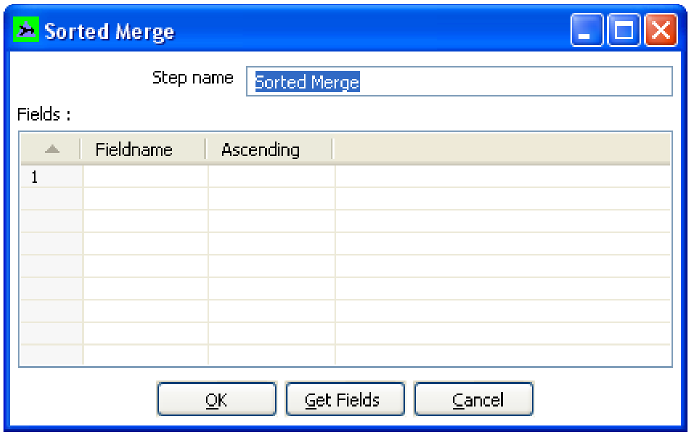
- “identical” : The key was found in both streams and the values to compare were identical;
- “changed” : The key was found in both streams but one or more values is different;
- “new” : The key was not found in the reference stream;
- “deleted” : The key was not found in the compare stream.

The row coming from the compare stream is passed on to the next steps, except for the “deleted” case.

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

11.6.43. Sorted Merge

11.6.43.1. Screenshot



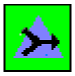
Sorted Merge

Step name: Sorted Merge

Fields :

	Fieldname	Ascending
1		

11.6.43.2. Icon



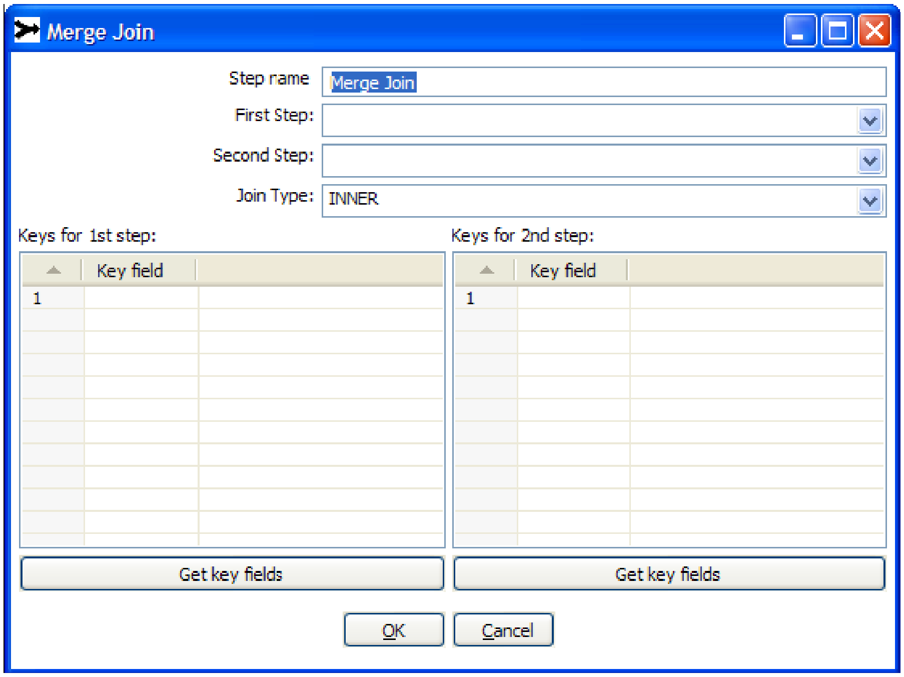
11.6.43.3. General description and usage

The Sorted Merge step merges rows coming from multiple input steps providing these rows are sorted themselves on the given key fields. Options include:

- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **Fields**: Specify fields on which the incoming data is sorted and the sort order

11.6.44. Merge Join

11.6.44.1. Screenshot



Step name: Merge Join

First Step:

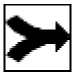
Second Step:

Join Type: INNER

Keys for 1st step:

Keys for 2nd step:

11.6.44.2. Icon



11.6.44.3. General description and usage

The Merge Join step performs a classic merge join between data sets with data coming from two different input steps. Join options include INNER, LEFT OUTER, RIGHT OUTER, and FULL OUTER.

NOTE: *This step expects the rows coming in to be sorted on the specified key fields*

11.6.44.4. Options

- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **First Step**: Specify the first input step to the merge join
- **Second Step**: Specify the second input step to the merge join
- **Join Type**: Select from the available types of joins
- **Key fields**: Specify the key fields on which the incoming data is sorted
- **Get key fields** buttons: Retrieves a list of fields from the specified first and second steps

11.6.45. Java Script Value

11.6.45.1. Screenshot

11.6.45.2. Icon

11.6.45.3. General description

This step type allows you to do complex calculations using the JavaScript language. The JavaScript engine used is Rhino 1.5R5.

11.6.45.4. Options

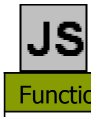
- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **Java Script**: The script in question.
- **Fields**: These are the fields that we want to add to the output stream.

11.6.45.5. 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.

11.6.45.6. Value functions

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



Function	Description
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 string 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.*

Function	Description
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.

Function	Description
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.

Function	Description
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 10.4.1.5.1. 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 10.4.1.5.1. Number formats Arg1: localized date-time pattern characters (u, t)

Function	Description
Value num2dat()	Converts a number to a date based upon the number of milliseconds since January 1st, 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 10.4.1.5.1. 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 10.4.1.5.1. 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 1st, 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 p
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, 0=miliseconds
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.

11.6.45.7. JavaScript Examples

11.6.45.7.1. 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.

11.6.45.7.2. Sets the location name of an address to uppercase

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

11.6.45.7.3. Extract information from a date field

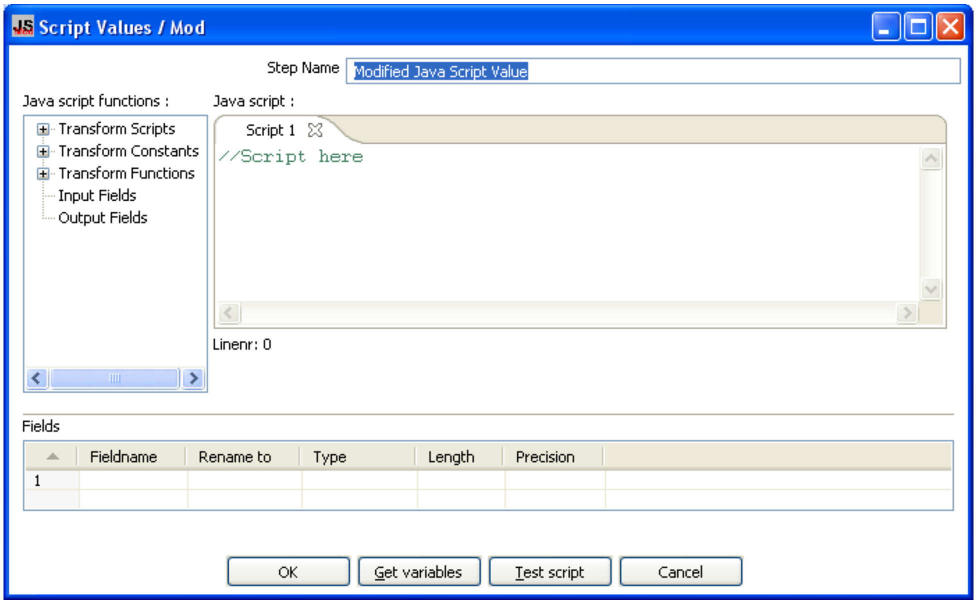
```
// Year/Month/Day representation:
ymd = date_field.Clone().dat2str("yyyy/MM/dd").getString();
// Day/Month/Year representation:
dmy = date_field.Clone().dat2str("dd/MM/yyyy").getString();
// Year/Month :
ym = date_field.Clone().dat2str("yyyy/MM").getString();
// Long description of the month in the 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: 10.4.1.5.1. Number formats

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

11.6.46. Modified Java Script Value

11.6.46.1. Screenshot



Step Name: Modified Java Script Value

```
//Script here
```

Linenr: 0

	Fieldname	Rename to	Type	Length	Precision
1					

11.6.46.2. Icon

11.6.46.3. General Description

This is a modified version of the "Javascript Values" step that provides better performance and an easier, expression based user interface for building javascript expressions. This step will also allow you to create multiple scripts for each step.

11.6.46.4. Java script functions

This section provides a tree view of your available scripts, functions, input fields and output fields.

- **Transformation Scripts**: displays a list of scripts you have created in this step
- **Transformation Constants**: a list of pre-defined, static constants including SKIP_TRANSFORMATION, ERROR_TRANSFORMATION, and CONTINUE_TRANSFORMATION
- **Transformation Functions**: contains a variety of String, Numeric, Date, Logic and specialized functions you can use to create your script. To add a function to your script, simply double-click on the function or drag it to the location in your script that you wish to insert it.
- **Input Fields**: a list of inputs coming into the step. Double-click or use drag and drop to insert the field into your script.
- **Output Fields**: a list of outputs for the step.

11.6.46.5. Java Script

This section is where you edit the script for this step. You can insert functions, constants, input fields, etc. from the tree control on the left by double-clicking on the node you wish to insert or by dragging the object onto the Java Script panel.

11.6.46.6. Fields

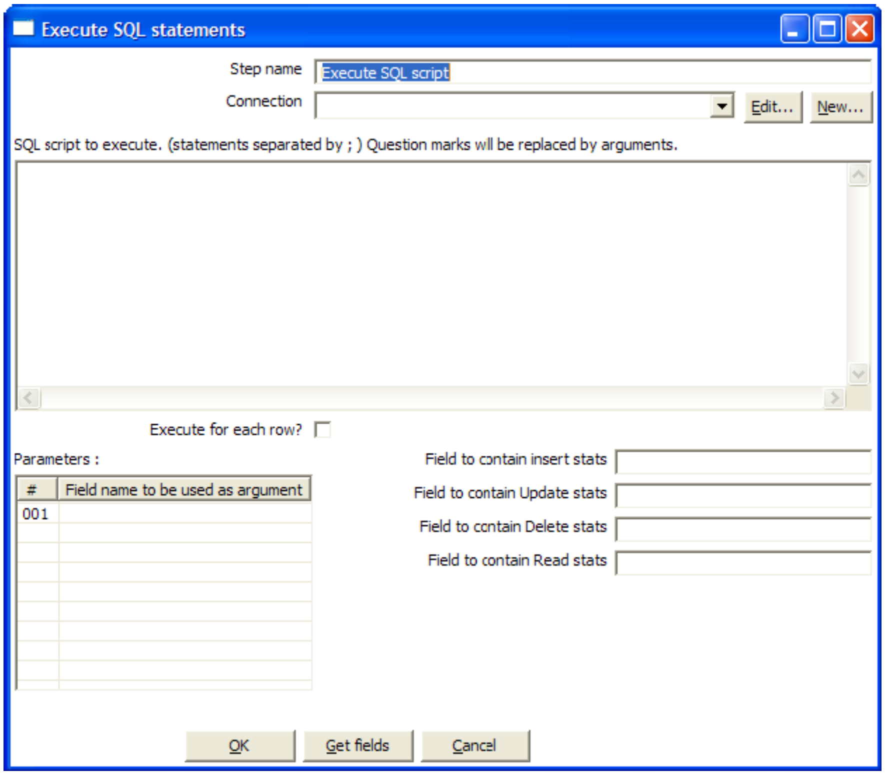
The Fields table contains a list of variables from your script including the ability to add metadata like a descriptive name.

11.6.46.7. Extras

- **Get Variables**: Retrieves a list of variables from your script.
- **Test script**: Use this button to test the syntax of your script.

11.6.47. Execute SQL script

11.6.47.1. Screenshot



Execute SQL statements

Step name: Execute SQL script

Connection

SQL script to execute. (statements separated by ;) Question marks will be replaced by arguments.

Execute for each row? □

Parameters :

#	Field name to be used as argument
001	

Field to contain insert stats

Field to contain Update stats

Field to contain Delete stats

Field to contain Read stats

11.6.47.2. Icon

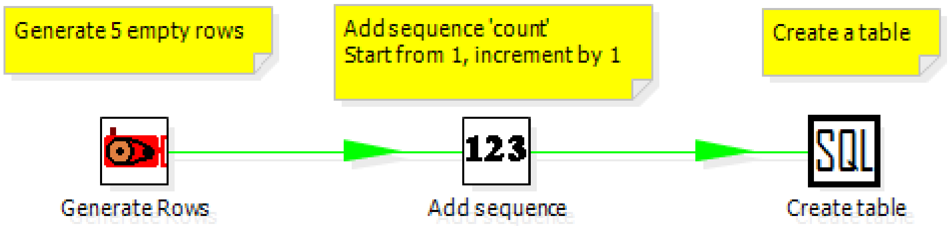
SQL

11.6.47.3. General description

You can execute SQL scripts with this step, either once, during the initialization 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:

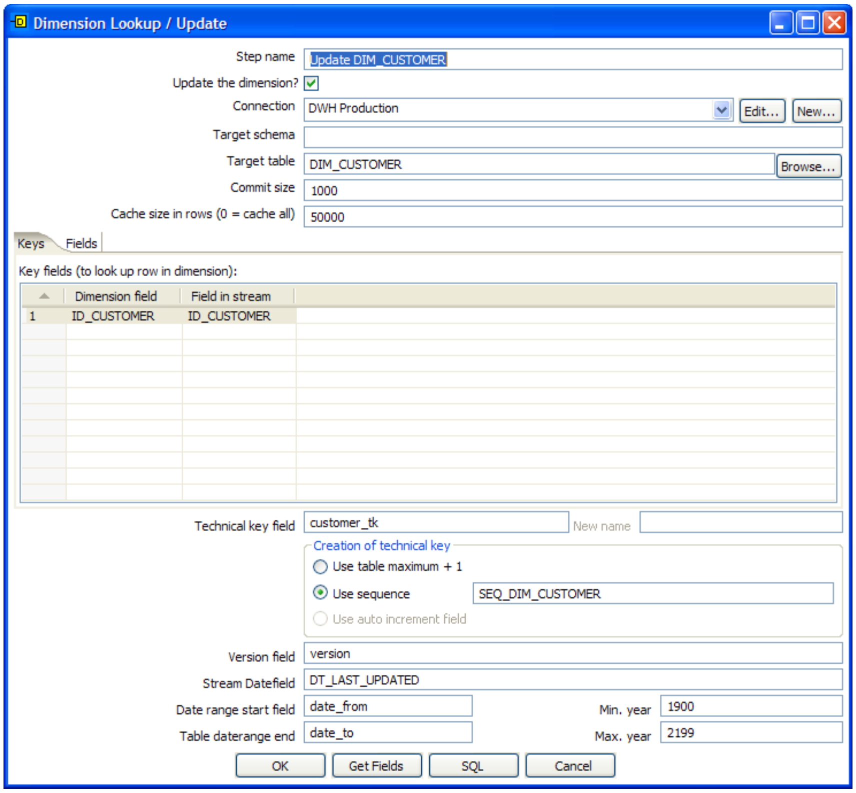
```
CREATE TABLE tab?
(
a INTEGER
);
```

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.

11.6.48. Dimension lookup/update

11.6.48.1. Screen shot



Dimension Lookup / Update

Step name: Update DIM_CUSTOMER

Update the dimension? ☑

Connection: DWH Production

Target schema:

Target table: DIM_CUSTOMER

Commit size: 1000

Cache size in rows (0 = cache all): 50000

Key fields (to look up row in dimension):

#	Dimension field	Field in stream
1	ID_CUSTOMER	ID_CUSTOMER

Technical key field: customer_tk

New name:

Creation of technical key

○ Use table maximum + 1

● Use sequence: SEQ_DIM_CUSTOMER

○ Use auto increment field

Version field: version

Stream Datefield: DT_LAST_UPDATED

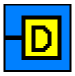
Date range start field: date_from

Min. year: 1900

Table daterange end: date_to

Max. year: 2199

11.6.48.2. Icon



11.6.48.3. General description

This step type allows you to implement Ralph Kimball's slowly changing dimension for both types: Type I (update) and Type II (insert). 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.*

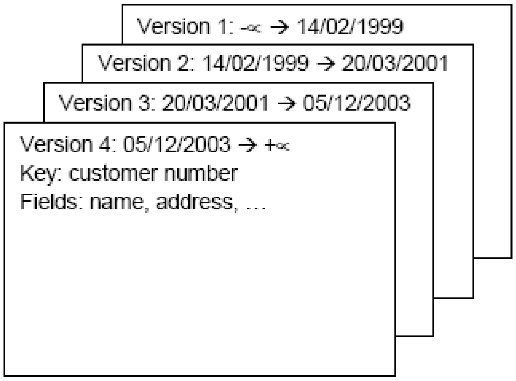
11.6.48.4. Options

- **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.
- **Target Schema**: This allows you to specify a schema name to improve precision in the quoting and allow for table-names with dots '.' in it.
- **Commit size**: Setting this to 10 will generate a commit every 10 inserts or updates.
- **Cache size:** this is the cache size in number of rows that will be held in memory to speed up lookups by reducing the number of round trips to the database. Please note that only the last version of a dimension entry is kept in memory. If there are more entries passing than what can be kept in memory, the technical keys with the highest values are kept in memory in the hope that these are the most relevant.
  A cache size of 0 caches as many rows as possible and until your JVM runs out of memory. Use this option wisely with dimensions that can't grown too large.
  A cache size of -1 means that caching is disabled.
- **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 I 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.

11.6.48.5. 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.


Version 1: -∝ → 14/02/1999
Version 2: 14/02/1999 → 20/03/2001
Version 3: 20/03/2001 → 05/12/2003
Version 4: 05/12/2003 → +∝
Key: customer number
Fields: name, address, …

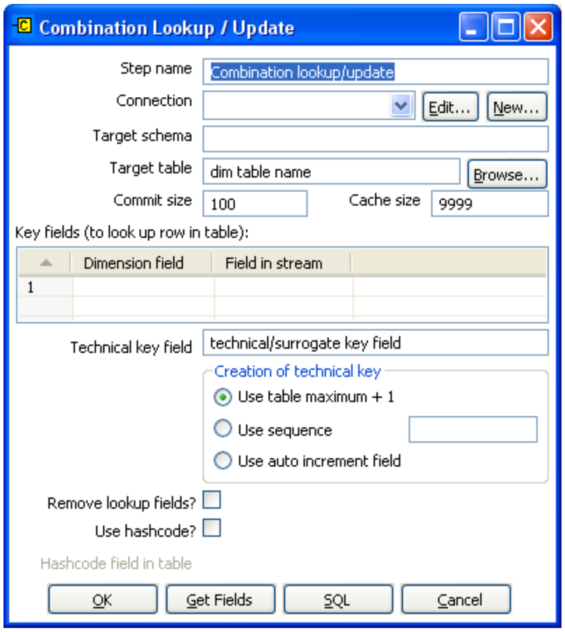
Each dimension entry can be represented by a stack of papers containing the information valid during a certain period of time.
Insert then means that we add a new piece of paper containing the new information. (Type II)
Punch through means that we overwrite certain data on all pieces of paper for that certain customer number. (Type I)

11.6.48.6. 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.
  IMPORTANT: this needs to be a Date field. We isolate functionality and as such require you to do date type conversions in advance.
- **For the "Date range start and end fields"**: You can only enter 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.

11.6.49. Combination lookup/update

11.6.49.1. Screenshot



Combination Lookup / Update

Step name: Combination lookup/update

Connection:

Target schema:

Target table: dim table name

Commit size: 100

Cache size: 9999

Key fields (to look up row in table):

	Dimension field	Field in stream
1		

Technical key field: technical/surrogate key field

Creation of technical key

● Use table maximum + 1

○ Use sequence

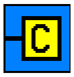
○ Use auto increment field

Remove lookup fields? □

Use hashcode? □

Hashcode field in table

11.6.49.2. Icon



11.6.49.3. 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... fieldn 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.

11.6.49.4. 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 schema**: The name of the Schema for the table to write data to. This is important for data sources that allow for table names with dots '.' In it.
- **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 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.).
- **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. IMPORTANT: This hash code is NOT unique. As such it makes no sense to place a unique index on it.

11.6.49.5. Extra

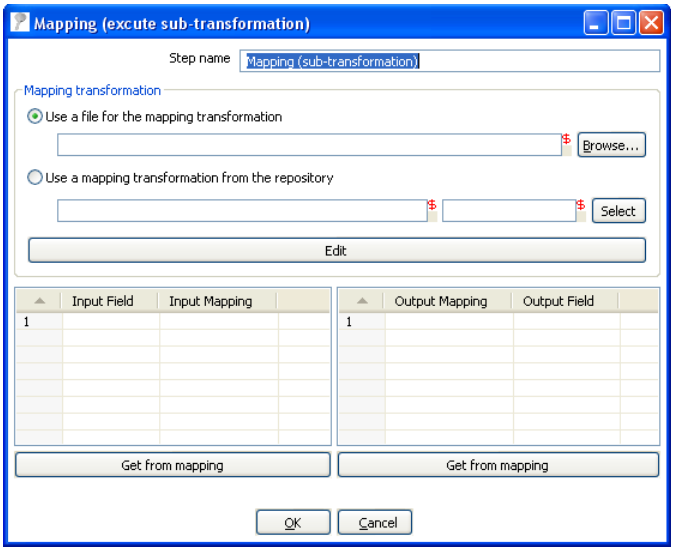
- **Get fields**: Fills in all the available fields on the input stream, except for the keys you specified.
- **SQL**: Generates the SQL to build the dimension and allows you to execute this SQL.

11.6.49.6. 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.

11.6.50. Mapping

11.6.50.1. Screenshot



Mapping (excute sub-transformation)

Step name: Mapping (sub-transformation)

Mapping transformation

● Use a file for the mapping transformation

○ Use a mapping transformation from the repository

11.6.50.2. Icon

11.6.50.3. 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

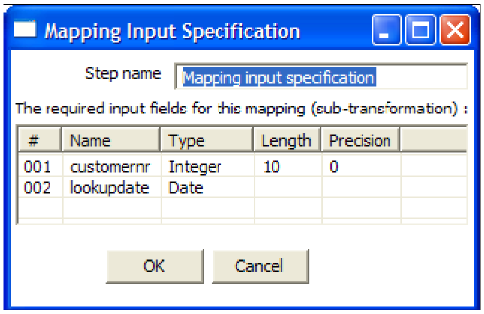
– specified how the input fields are transformed: the field that are added and deleted

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 mapping 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 inputs we specify in the mapping input step:

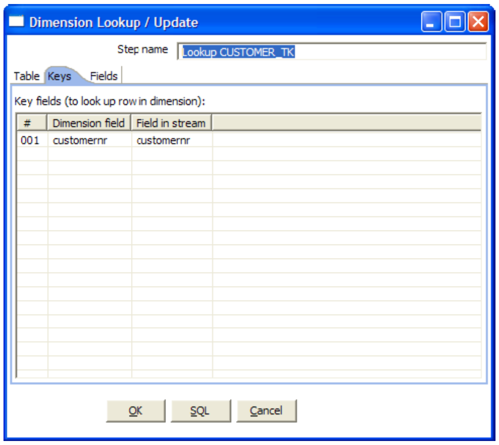


Step name: Mapping input specification

The required input fields for this mapping (sub-transformation) :

#	Name	Type	Length	Precision
001	customernr	Integer	10	0
002	lookupdate	Date		

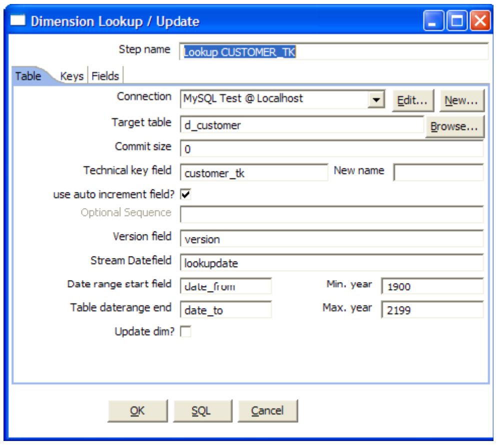
After this we can perform any calculation in our reusable transformation (Mapping), in our case we do a lookup in the dimension:



Step name: Lookup CUSTOMER_TK

Key fields (to look up row in dimension):

#	Dimension field	Field in stream
001	customernr	customernr



Step name: Lookup CUSTOMER_TK

Connection: MySQL Test @ Localhost

Target table: d_customer

Commit size: 0

Technical key field: customer_tk New name:

use auto increment field? ☑

Optional Sequence:

Version field: version

Stream Datefield: lookupdate

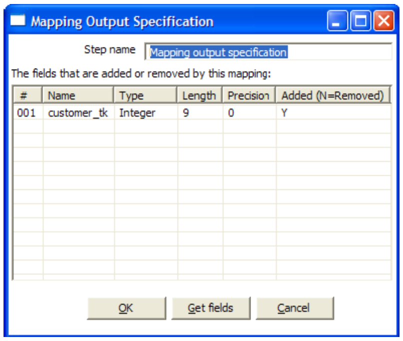
Date range start field: date_from Min. year: 1900

Table daterange end: date_to Max. year: 2199

Update dim? □

This dimension lookup step adds one field to the equation: customer_tk.

We can specify the fields that were in the Mapping Output step:

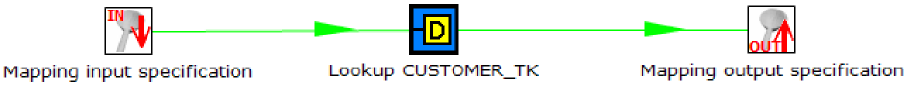


Step name: Mapping output specification

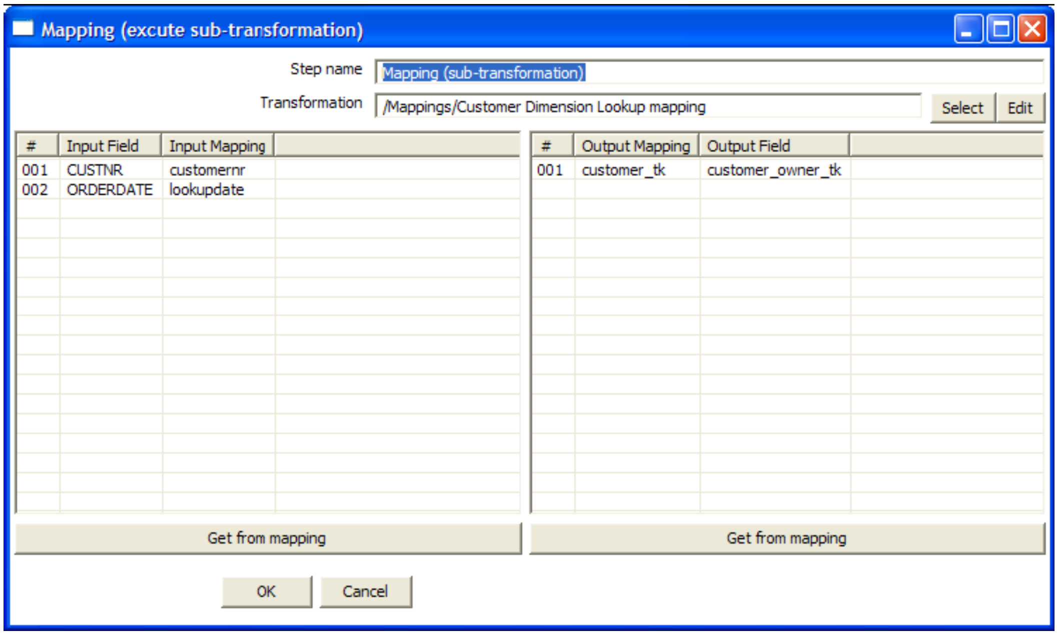
The fields that are added or removed by this mapping:

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

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/Customer Dimension Lookup mapping

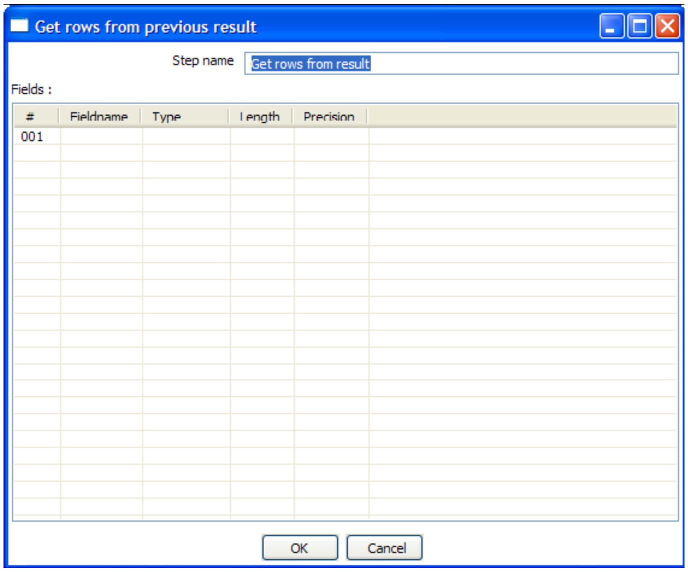
#	Input Field	Input Mapping
001	CUSTNR	customernr
002	ORDERDATE	lookupdate

#	Output Mapping	Output Field
001	customer_tk	customer_owner_tk

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.

11.6.51. Get rows from result

11.6.51.1. Screenshot



Step name: Get rows from result

Fields :

#	Fieldname	Type	Length	Precision
001				

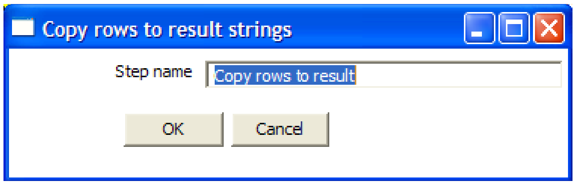
11.6.51.2. Icon

11.6.51.3. 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.
IMPORTANT: no validation of the supplied metadata is done at this time to allow for greater flexibility. It is just an aid at design time.

11.6.52. Copy rows to result

11.6.52.1. Screenshot



Step name: Copy rows to result

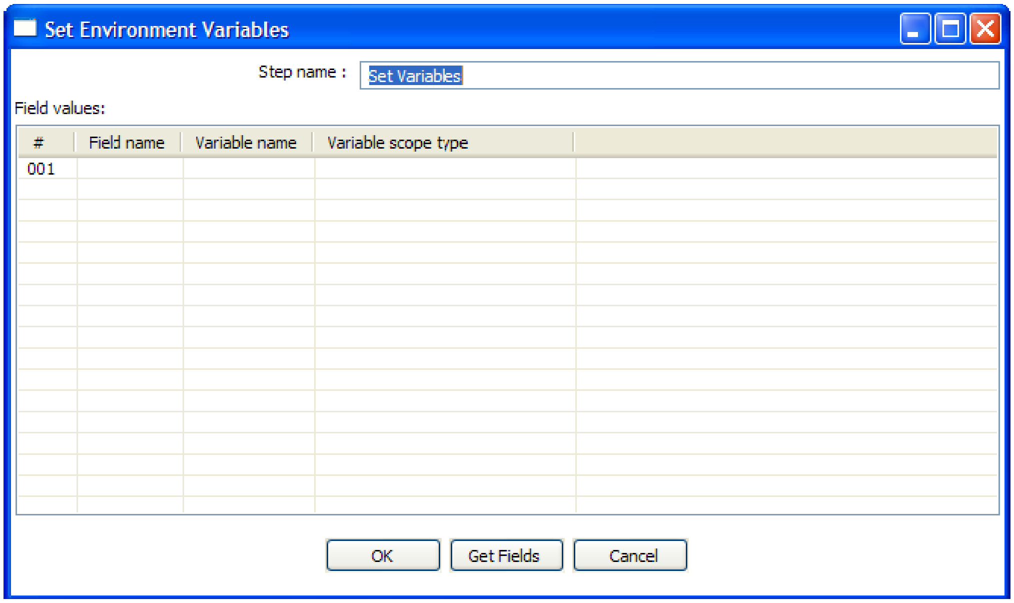
11.6.52.2. Icon

11.6.52.3. General description

This step allows you to transfer rows of data (in memory) to the next transformation (job entry) in a job.

11.6.53. Set Variable

11.6.53.1. Screenshot



Step name : Set Variables

Field values:

#	Field name	Variable name	Variable scope type
001			

11.6.53.2. Icon

11.6.53.3. 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:

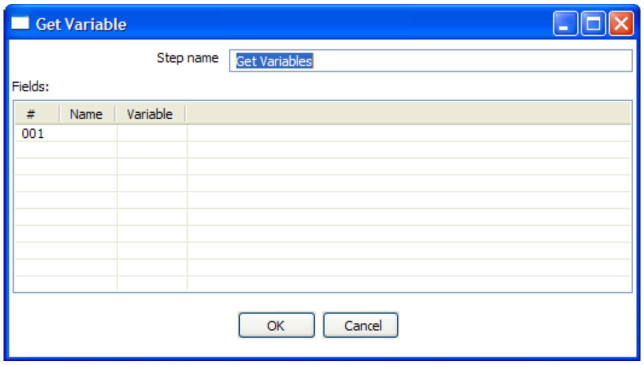
- **Valid in the virtual machine**: the complete virtual machine will know about this variable. **WARNING**: this makes your transformation only fit to run in a stand-alone fashion. Running on an application server like on the Pentaho framework can become a problem. That is because other transformations 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.

11.6.53.4. Variable usage

Refer to 8.1 Variables for a description of the use of variables.

11.6.54. Get Variable

11.6.54.1. Screenshot



Step name: Get Variables

Fields:

#	Name	Variable
001		

11.6.54.2. Icon

11.6.54.3. General description

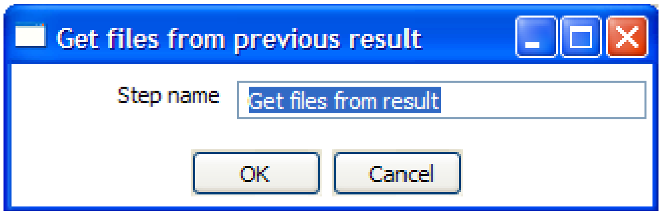
This step allows you to get the value of a variable. This step can return rows or add values to input rows.

NOTE: *You need to specify the complete variable specification in the format* `${variable}` *or* `%%variable%%` *(as described in 8.1 Variables) . That means you can also enter complete strings in the variable column, not just a variable.*

For example, you can specify: `${java.io.tmpdir}/kettle/tempfile.txt` and it will be expanded to /tmp/kettle/tempfile.txt on Unix-like systems.

11.6.55. Get files from result

11.6.55.1. Screenshot



Step name: Get files from result

11.6.55.2. Icon

Get Files

11.6.55.3. General description

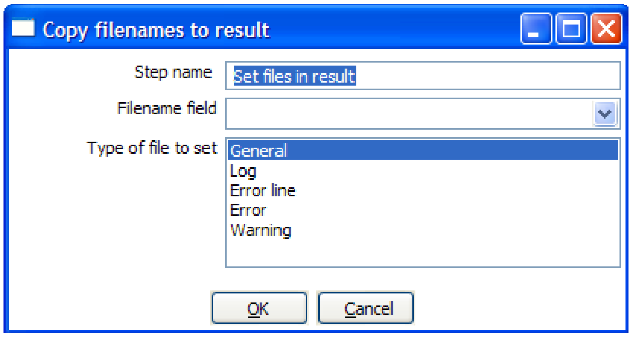
Every time a file gets processed, used or created in a transformation or a job, the details of the file, the job entry, the step, etc. is captured and added to the result. You can access this file information using this step.

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

11.6.56. Set files in result

11.6.56.1. Screenshot



Copy filenames to result

Step name: Set files in result

Filename field:

Type of file to set: General (selected), Log, Error line, Error, Warning

11.6.56.2. Icon

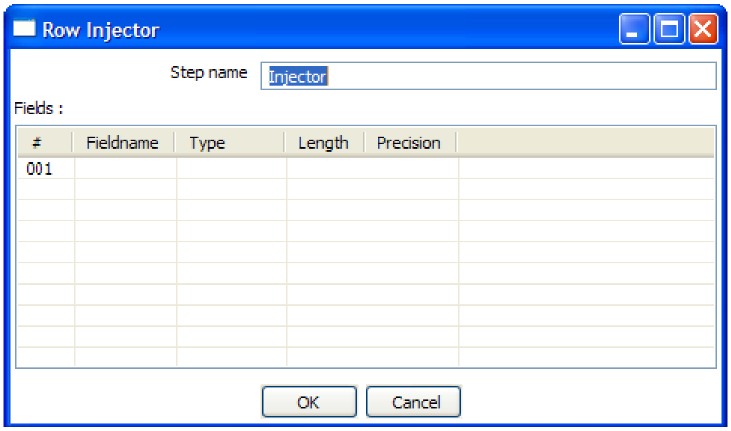
11.6.56.3. General description

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.

11.6.57. Injector

11.6.57.1. Screenshot

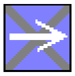


Step name: Injector

Fields :

#	Fieldname	Type	Length	Precision
001				

11.6.57.2. Icon



11.6.57.3. 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 tings 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);
```

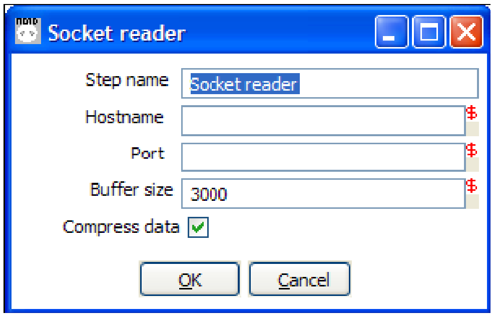
- After that you start the threads in the transformation. Then you can inject the rows while the transformation is running:

```
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.

11.6.58. Socket reader

11.6.58.1. Screen shot



Step name: Socket reader

Hostname:

Port:

Buffer size: 3000

Compress data ☑

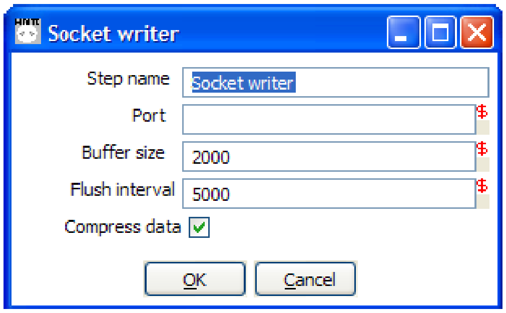
11.6.58.2. Icon

11.6.58.3. General description and use

Socket readers are used to transfer data from one server to another over TCP/IP. The primary use for these steps is in-line in a clustering environment. If you want to use these yourself, make sure to synchronize the preparation and start cycles of the transformations between the hosts. (like the clustered transformation does)

11.6.59. Socket writer

11.6.59.1. Screen shot



Step name: Socket writer

Port:

Buffer size: 2000

Flush interval: 5000

Compress data ☑

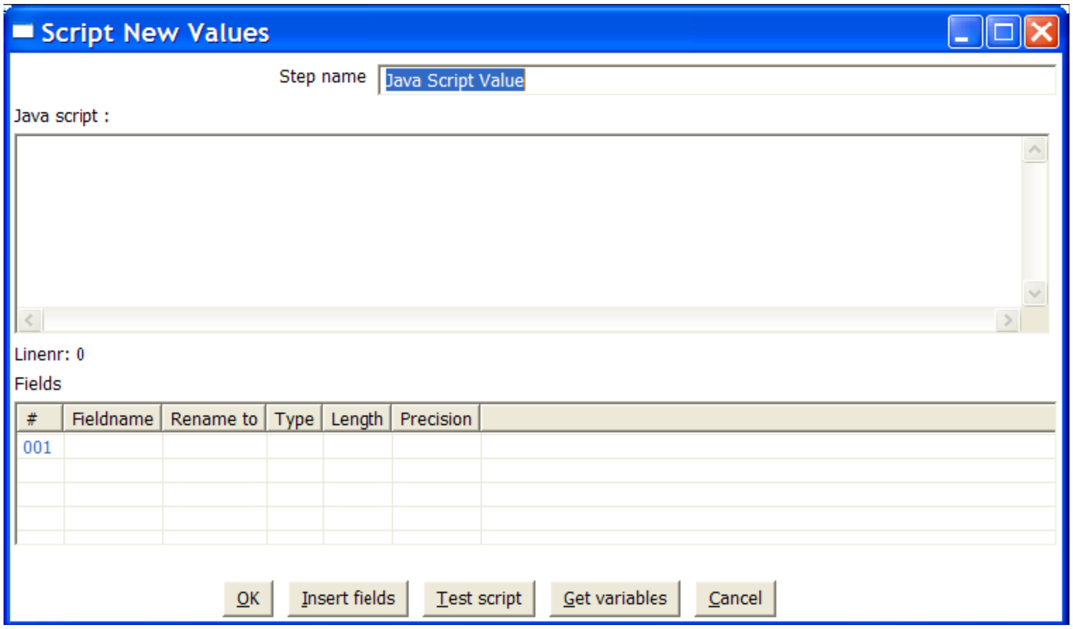
11.6.59.2. Icon

11.6.59.3. General description and use

Socket writers are used to transfer data from one server to another over TCP/IP. The primary use for these steps is in-line in a clustering environment. If you want to use these yourself, make sure to synchronize the preparation and start cycles of the transformations between the hosts. (like the clustered transformation does)

11.6.60. Aggregate Rows

11.6.60.1. Screenshot



Script New Values

Step name: Java Script Value

Java script :

Linenr: 0

Fields

#	Fieldname	Rename to	Type	Length	Precision
001					

11.6.60.2. Icon

Σ

11.6.60.3. 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

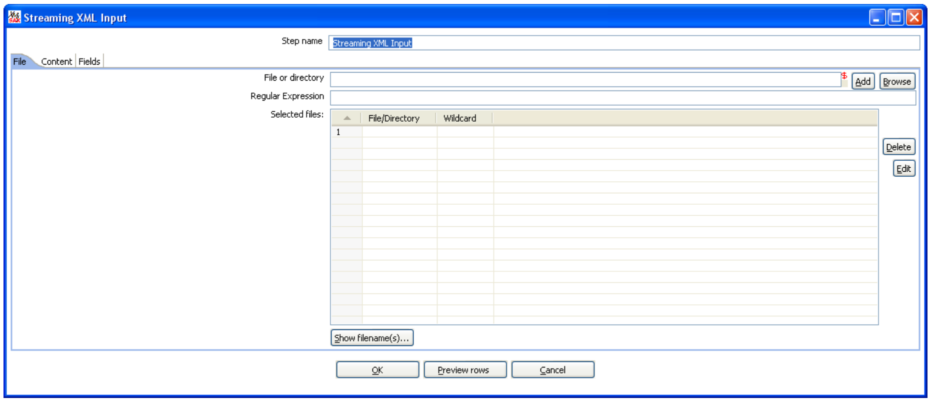
NOTE: *This step type is deprecated. See the "Group By" step in 11.4.32. Group By for a more powerful way of aggregating rows of data. The aggregate step can be removed in a future version.*

11.6.60.4. Options

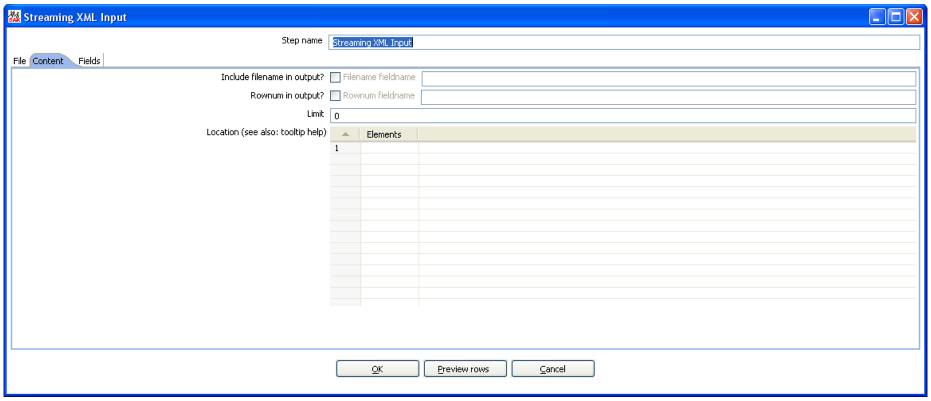
- **Step name**: Name of the step. This name has to be unique in a single transformation.
- **Fields**: The fields and the aggregation type.

11.6.61. Streaming XML Input

11.6.61.1. Screenshot



Step name: Streaming XML Input

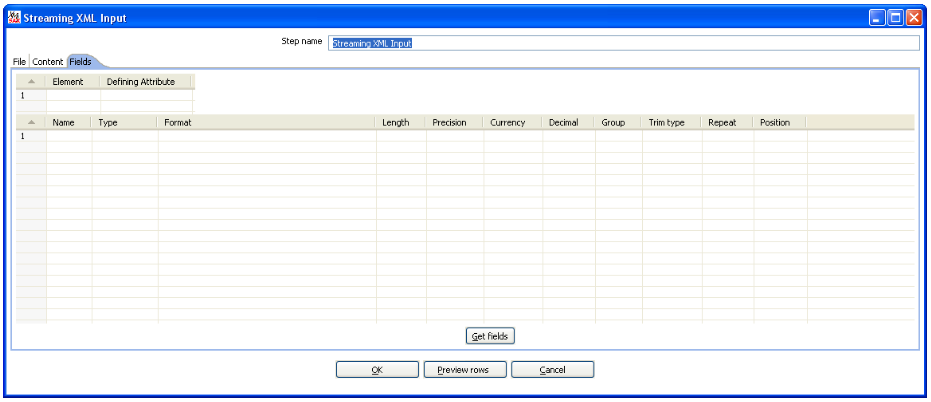


Step name: Streaming XML Input

Limit: 0

The "File" tab

The "Content" tab



Step name: Streaming XML Input

The "Fields" tab

11.6.61.2. Icon

11.6.61.3. General description

11.6.61.4. The purpose of this step is to provide value parsing. This step is based on SAX parser to provide better performances with larger files. It is very similar to Xml Input, there are only differences in content and field tabs.

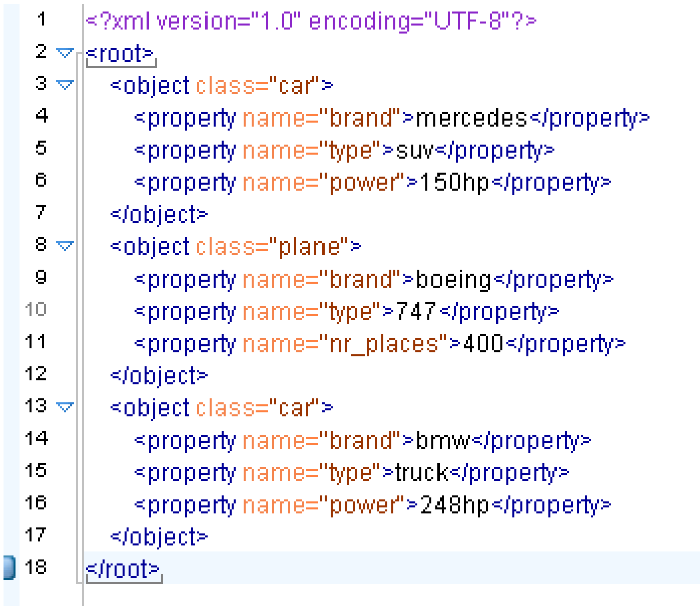
11.6.61.5. Options

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. **NOTE**: *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.
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.
Limit	You can specify the maximum number of rows to read here.
Location	Specify the path by way of elements to the repeating part of the XML file. The element column is used to specify the element and position as follows: **A**: still specify an attribute **Ep**: specify an element defined by position (equivalent to E in original XMLInput). **Ea**: specify an element defined by an attribute and allow value parsing. **Example:** `Ep=element/1` this is the first element called "element" `Ea=element/att:val` this is the element called "element" that have an attribute called "att" with "val" value
Fields	• Name: Name of the field. • Type: Type of the field can be either String, Date or Number. • Format mask to convert with: See 8.5.1.5 for a complete description of format specifiers. • Length: o Number: Total number of significant figures in a number; o String: total length of string; o Date: length of printed output of the string (e.g. 4 only gives back the year). • Precision: o Number: Number of floating point digits; o String: unused;

Option	Description
	○ Date: unused. • Currency: Symbol used to represent currencies 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 "," (10,000.00) or "." (5.000,00) • Trim type: The trimming method to apply on the string found in the XML. • Repeat: Check this if you want to repeat empty values with the corresponding value from the previous row. • Position: The position of the XML element or attribute. You use the following syntax to specify the position of an element: The first element called "element": `E=element/1` The first attribute called "attribute": `A=attribute/1` The first attribute called "attribute" in the second "element" tag: `E=element/2, A=attribute/1` **NOTE**: *You can auto-generate all the possible positions in the XML file supplied by using the "Get Fields" button.* **NOTE**: *Support was added for XML documents where all the information is stored in the Repeating (or Root) element. The special R= locater was added to allow you to grab this information. The "Get fields" button finds this information if it's present.*

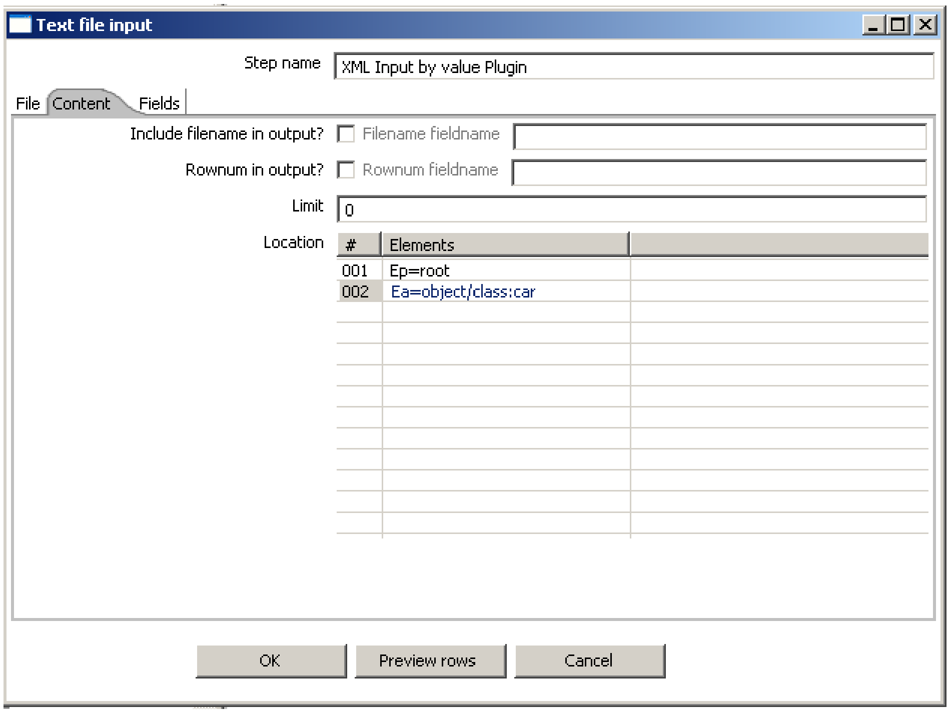
11.6.61.6. Complete example

Consider the following XML:



```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <object class="car">
    <property name="brand">mercedes</property>
    <property name="type">suv</property>
    <property name="power">150hp</property>
  </object>
  <object class="plane">
    <property name="brand">boeing</property>
    <property name="type">747</property>
    <property name="nr_places">400</property>
  </object>
  <object class="car">
    <property name="brand">bmw</property>
    <property name="type">truck</property>
    <property name="power">248hp</property>
  </object>
</root>
```

Suppose that we are interested in cars we must specify the location of the repeating element like this:



Step name: XML Input by value Plugin

Include filename in output? ☐ Filename fieldname

Rownum in output? ☐ Rownum fieldname

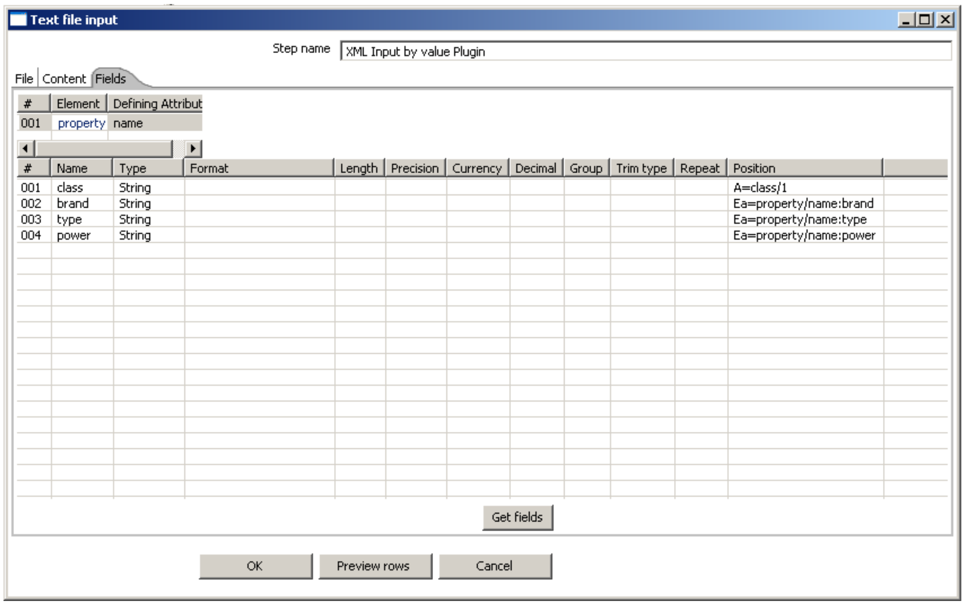
Limit: 0

Location

#	Elements
001	Ep=root
002	Ea=object/class:car

Now lets see the fields, we have different "property" elements that are differentiated by their "name" attribute, we are about to have the following fields "brand", "type" and "power" according to the "name" attribute.

For this, we must specify the association between "property" and "name" in the first grid.



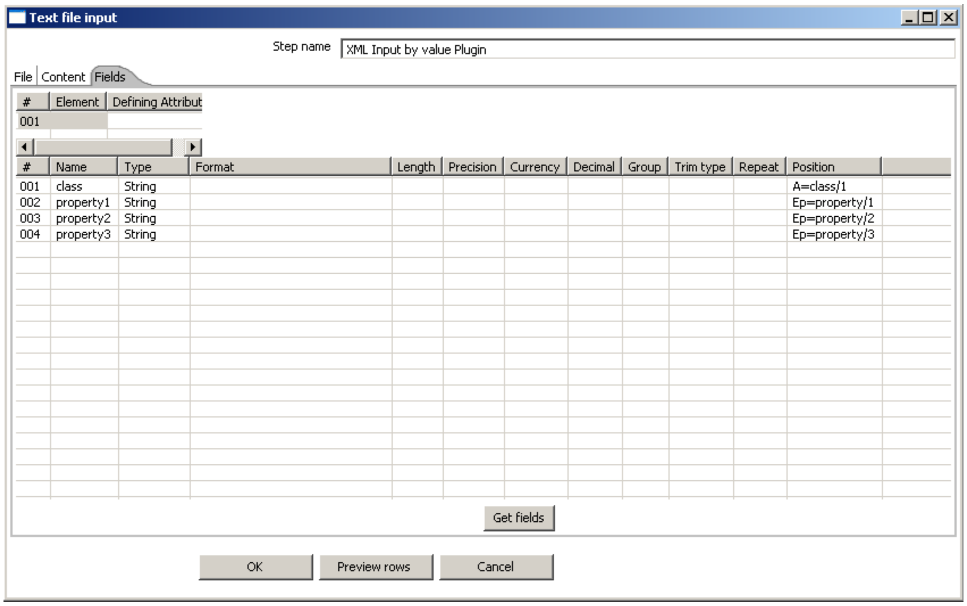
Step name: XML Input by value Plugin

#	Element	Defining Attribut
001	property	name

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type	Repeat	Position
001	class	String									A=class/1
002	brand	String									Ea=property/name:brand
003	type	String									Ea=property/name:type
004	power	String									Ea=property/name:power

Pressing the “Get Fields” button retrieves the right fields including properties.

Let us now try leaving the new grid empty.



Step name: XML Input by value Plugin

#	Element	Defining Attribut
001		

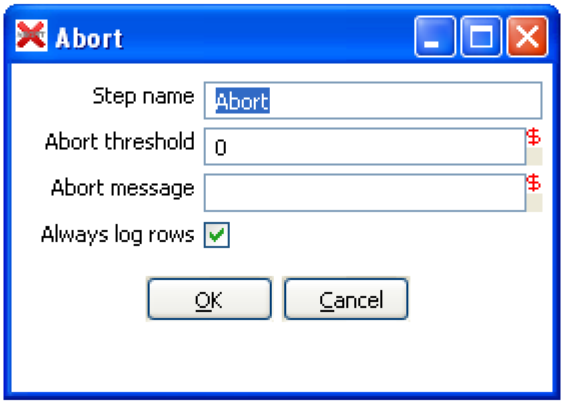
#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type	Repeat	Position
001	class	String									A=class/1
002	property1	String									Ep=property/1
003	property2	String									Ep=property/2
004	property3	String									Ep=property/3

You can see that in this case the step is working like the original XMLInput and retrieve fields by their position.

In this case, it is better to use value parsing, cause you get the right field names, and missing elements will not corrupt results (for example missing <property name="power"></property> in some rows).

11.6.62. Abort

11.6.62.1. Screenshot



Step name: Abort

Abort threshold: 0

Abort message:

Always log rows ☑

11.6.62.2. Icon

11.6.62.3. General description

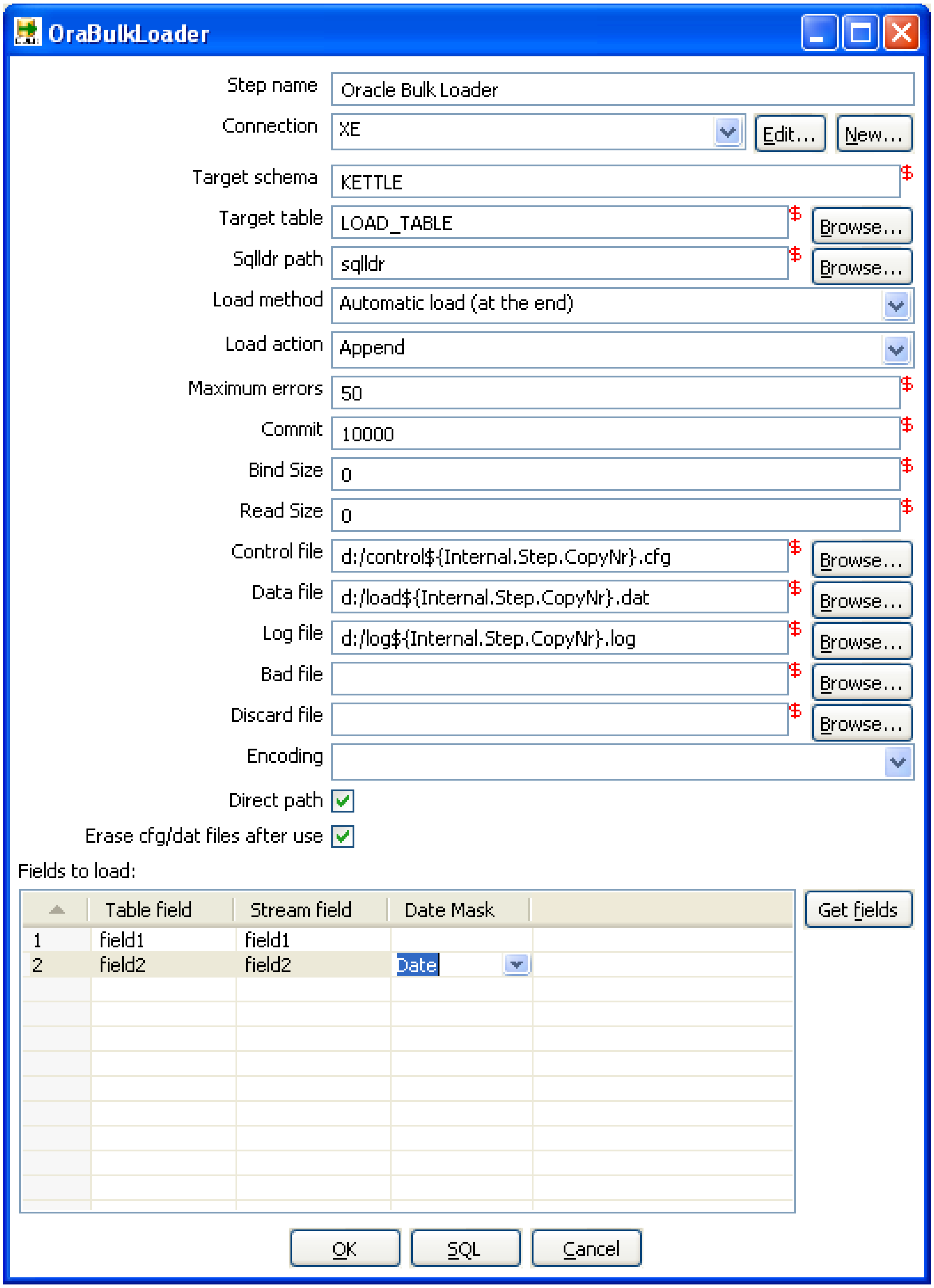
This step type allows you abort a transformation upon seeing input. It's main use is in error handling, so that a transformation e.g. can be aborted after x number of rows flow to over an error hop.

11.6.62.4. Options

- **Step name**: Name of the step. This name has to be unique in a single transformation;
- **Abort threshold**: The threshold of number of rows after which to abort the transformations. E.g. If threshold is 0, the abort step will abort after seeing the first row. If threshold is 5, the abort step will abort after seeing the sixth row;
- **Abort message**: The message to put in the log upon aborting. If not filled in a default message will be used;
- **Always log rows**: Always log the rows processed by the Abort step. This allows the rows to be logged although the log level of the transformation would normally not do it. This way you can always see in the log which rows caused the transformation to abort.

11.6.63. Oracle Bulk Loader

11.6.63.1. Screenshot



Step name: Oracle Bulk Loader

Connection: XE

Target schema: KETTLE

Target table: LOAD_TABLE

Sqlldr path: sqlldr

Load method: Automatic load (at the end)

Load action: Append

Maximum errors: 50

Commit: 10000

Bind Size: 0

Read Size: 0

Control file: d:/control${Internal.Step.CopyNr}.cfg

Data file: d:/load${Internal.Step.CopyNr}.dat

Log file: d:/log${Internal.Step.CopyNr}.log

Bad file:

Discard file:

Encoding:

Direct path ☑

Erase cfg/dat files after use ☑

Fields to load:

	Table field	Stream field	Date Mask
1	field1	field1	
2	field2	field2	Date

11.6.63.2. Icon

11.6.63.3. General description

This step type allows you bulk load data to an Oracle database. It will write the data it receives to a proper load format and will then invoke Oracle SQL*Loader to transfer it to the specified table.

IMPORTANT! *Just like all steps in the "Experimental" category, this step is not considered ready for production use by the author. In the specific case of the Oracle Bulk loader we lacked the time to do extensive testing on it. Your feedback is most welcome as always.*

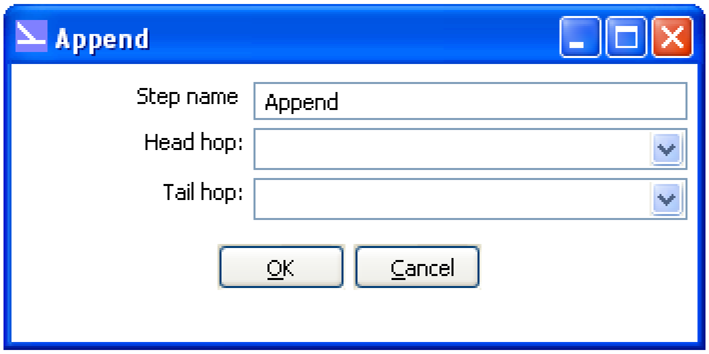
11.6.63.4. Options

- **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 schema**: The name of the Schema for the table to write data to. This is important for data sources that allow for table names with dots '.' In it;
- **Target table**: Name of the target table;
- **Sqlldr path**: Full path to the sqlldr utility (including sqlldr). If sqlldr is in the path of the executing application you can leave it to sqlldr;
- **Load method**: Either "Automatic load (at the end)" or "Manual load (only creation of files)". Automatic load will start up sqlldr after receiving all input with the specified arguments in this step. Manual load will only create a control and data file, this can be used as a back-door: you can have PDI generate the data and create e.g. your own control file to load the data (outside of this step).
- **Load action**: Append, Insert, Replace, Truncate. These map to the sqlldr action to be performed.
- **Maximum errors**: The number of rows in error after which sqlldr will abort. This corresponds to the "ERROR" attribute of sqlldr;
- **Commit**: The number of rows after which to commit, this corresponds to the "ROWS" attribute of sqlldr which differs between using a conventional and a direct path load;
- **Bind Size**: Corresponds to the "BINDSIZE" attribute of sqlldr;
- **Read Size**: Corresponds to the "READSIZE" attribute of sqlldr;
- **Control file**: The name of the file used as control file for sqlldr;
- **Data file**: The name of the data file in which the data will be written;
- **Log file**: The name of the log file, optionally defined;
- **Bad file**: The name of the bad file, optionally defined;
- **Discard file**: The name of the discard file, optionally defined;
- **Encoding**: Encodes data in a specific encoding, any valid encoding can be chosen besides the one in the drop down list;
- **Direct path**: Switch on direct path loading, corresponds to DIRECT=TRUE in sqlldr;
- **Erase cfg/dat files after use**: When switched on the control and data file will be erased after loading.

- **Table field**: Table field to be loaded in the Oracle table;
- **Stream field**: Field to be taken from the incoming rows;
- **Date mask**: Either "Date" or "Date mask", determines how date/timestamps will be loaded in Oracle. When left empty defaults to "Date" in case of dates.

11.6.64. Append

11.6.64.1. Screenshot



Step name: Append

Head hop:

Tail hop:

11.6.64.2. Icon

11.6.64.3. General description

This step type allows you to order the rows of 2 inputs hops. First all of the rows of the "Head hop" will be read and output, after that all of the rows of the "Tail hop" will be written to the output. If more than 2 hops need to be used you can use multiple append steps in sequence.

11.6.64.4. Options

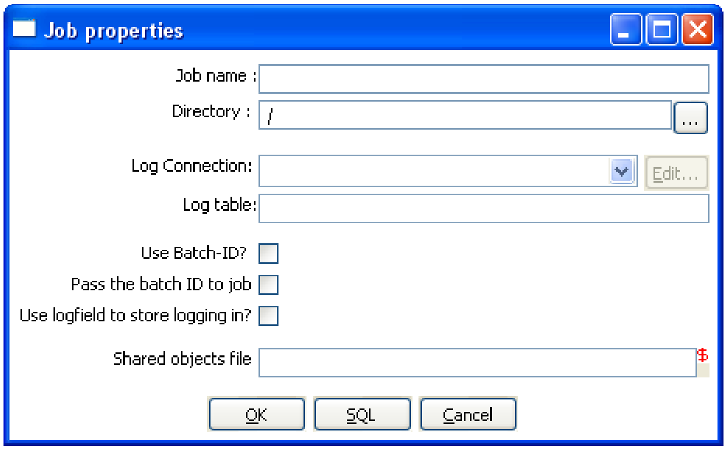
- **Step name**: Name of the step. This name has to be unique in a single transformation;
- **Head hop**: Name of the hop of which the rows should be output first;
- **Tail hop**: Name of the hop of which the rows should be output last.

12. Job Settings

12.1. Description

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

12.2. Screenshot



Job properties

Job name :

Directory : /

Log Connection:

Log table:

Use Batch-ID? □

Pass the batch ID to job □

Use logfield to store logging in? □

Shared objects file

12.3. Options

Option	Description
job name	the name of the job. This is required information if you want to save to a repository.
Directory	the directory in the repository in which you want to save the job.
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	Use a batch ID in the logging table
Pass the batch-ID to job entries?	Check this if you want to pass the generated unique batch ID to (transformation) job entries in this job.
Use logfield to store logging in?	Check this if you want to store the logging of this job in the logging table in a long text field. (CLOB)

12.4. Extra

SQL button: generates the SQL needed to create the logging table and allows you to execute this SQL statement.

13. Job Entries

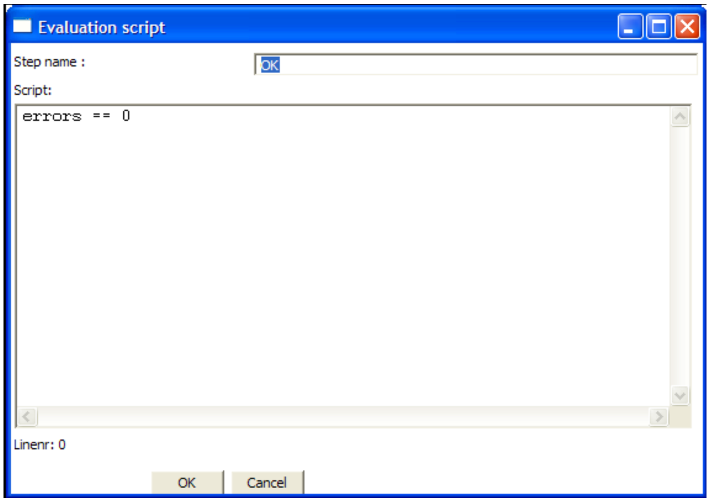
13.1. Description

A job entry is one part of a job. Job entries can provide you with a wide range of functionality ranging from executing transformations to getting files from a web server. Please see below for a complete list of all available job entry types.

13.2. Job Entry Types

13.2.1. Special Job Entries

13.2.1.1. Screenshot



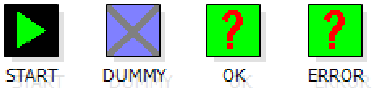
Step name : OK

Script:

```
errors == 0
```

Linenr: 0

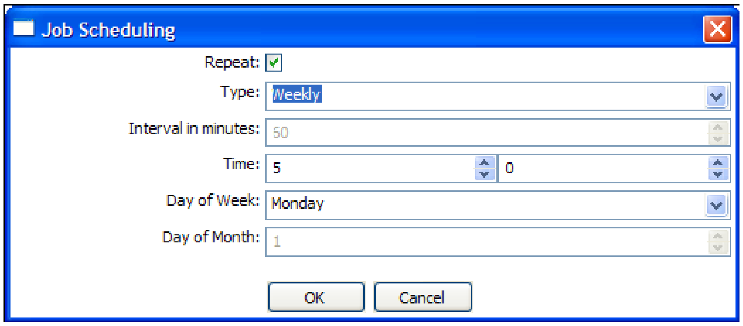
13.2.1.2. Icons



13.2.1.3. General description

13.2.1.3.1. Start

Start is where the job starts to execute and is required before the job can be executed.
Only unconditional job hops are available from a Start job entry.
The start icon also contains basic scheduling functionality. If you edit this job entry you'll get this screen:



Repeat: ☑

Type: Weekly

Interval in minutes: 50

Time: 5 0

Day of Week: Monday

Day of Month: 1

13.2.1.3.2. Dummy

Use the Dummy job entry to do nothing in a job. This can be useful to make job drawings clearer or for looping. Dummy performs no evaluation.

13.2.1.3.3. OK

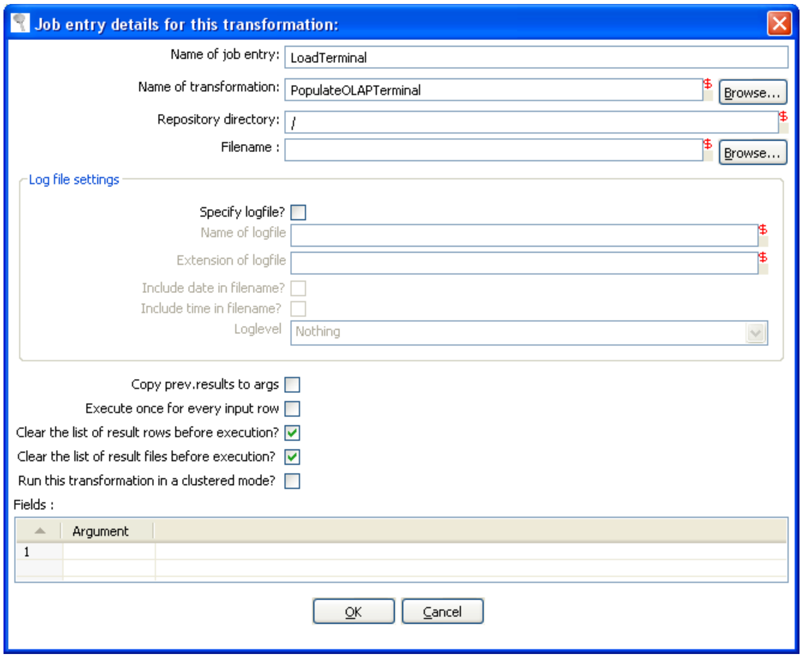
Checks if the number of errors from a previous job entry was 0 and results to true if this is the case.
See also the "Evaluation" job entry.

13.2.1.3.4. ERROR

Checks if the number of errors from a previous job entry was different from 0 and results to true if this is the case. See also the "Evaluation" job entry.

13.2.2. Transformation

13.2.2.1. Screenshot



Job entry details for this transformation:

Name of job entry: LoadTerminal

Name of transformation: PopulateOLAPTerminal

Repository directory: /

Filename :

Log file settings

Specify logfile? □

Name of logfile

Extension of logfile

Include date in filename? □

Include time in filename? □

Loglevel: Nothing

Copy prev.results to args □

Execute once for every input row □

Clear the list of result rows before execution? ☑

Clear the list of result files before execution? ☑

Run this transformation in a clustered mode? □

Fields :

	Argument
1	

13.2.2.2. Icon

13.2.2.3. General description

You can use the Transformation job entry to execute a previously defined transformation.

13.2.2.4. Options

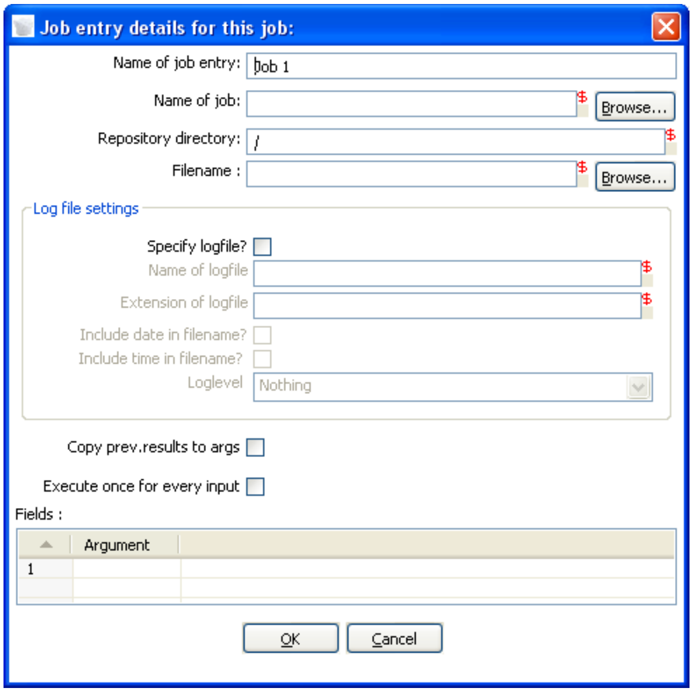
Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Name of transformation	The name of the transformation to start.
Repository directory	The directory in the repository where the transformation is located.
Filename	If you're not working with a repository, specify the XML filename of the transformation to start.
Specify log file	Check this if you want to specify a separate logging file for the execution of this transformation.
Name of log file	The directory and base name of the log file (for example C:\logs)
Extension of the log file	The filename extension (for example: log or txt)
Include date in filename	Adds the system date to the filename. (_20051231)
Include time in filename	Adds the system time to the filename. (_235959)
Logging level	Specifies the logging level for the execution of the transformation. See also the logging window in 15. Logging

Option	Description
Copy previous results to arguments .	The results from a previous transformation can be sent to this one using the "Copy rows to result" step
Arguments	Specify the strings to use as arguments for the transformation.
Execute once for every input row	Support for "looping" has been added by allowing a transformation to be executed once for every input row.
Clear the list or result rows before execution	Checking this makes sure that the list or result rows is cleared before the transformation is started.
Clear the list of result files before execution	Checking this makes sure that the list or result files is cleared before the transformation is started.

NOTE: *you can use variables ${path} in the filename and transformation name fields to specify the transformation to be executed.*

13.2.3. Job

13.2.3.1. Screenshot



13.2.3.2. Icon

13.2.3.3. General description

You can use the Job job entry to execute a previously defined job.

WARNING! *Although it is possible to create a recursive, never ending job that points to itself, you should be aware. This job will probably eventually fail with an out of memory or stack error.*

13.2.3.4. Options

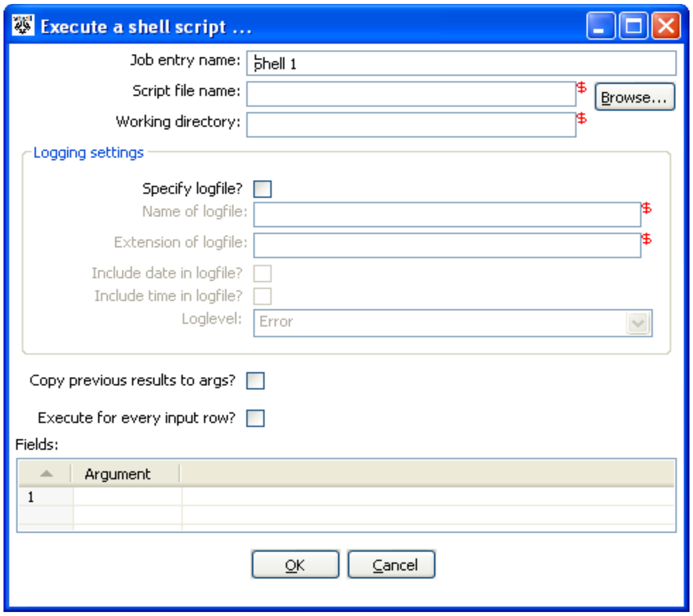
Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Name of transformation	The name of the job to start.
Repository directory	The directory in the repository where the job is located.
Filename	If you're not working with a repository, specify the XML filename of the job to start.
Specify log file	Check this if you want to specify a separate logging file for the execution of this job.
Name of log file	The directory and base name of the log file (for example C:\logs)
Extension of the log file	The filename extension (for example: log or txt)

Option	Description
Include date in filename	Adds the system date to the filename. (_20051231)
Include time in filename	Adds the system time to the filename. (_235959)
Logging level	Specifies the logging level for the execution of the job. See also the logging window in 15. Logging
Copy previous results to arguments	The results from a previous transformation can be sent to this job using the "Copy rows to result" step in a transformation.
Arguments	Specify the strings to use as arguments for the job.
Execute once for every input row	This implements looping. If the previous job entry returns a set of result rows, you can have this job executed once for every row found. One row is passed to this job at every execution. For example you can execute a job for each file found in a directory using this option.

NOTE: *you can use variables ${path} in the filename and job name fields to specify the job to be executed.*

13.2.4. Shell

13.2.4.1. Screenshot



Execute a shell script ...

Job entry name: Shell 1

Script file name:

Working directory:

Logging settings

Specify logfile? □

Name of logfile:

Extension of logfile:

Include date in logfile? □

Include time in logfile? □

Loglevel: Error

Copy previous results to args? □

Execute for every input row? □

Fields:

	Argument
1	

13.2.4.2. Icon

13.2.4.3. General description

You can use the Shell job entry to execute a shell script on the host where the job is running.

NOTE: *Shell scripts can (since release 2.3.0) output text to the console window. This output will be transferred to the Kettle logging system. Doing this no longer blocks the shell script.*

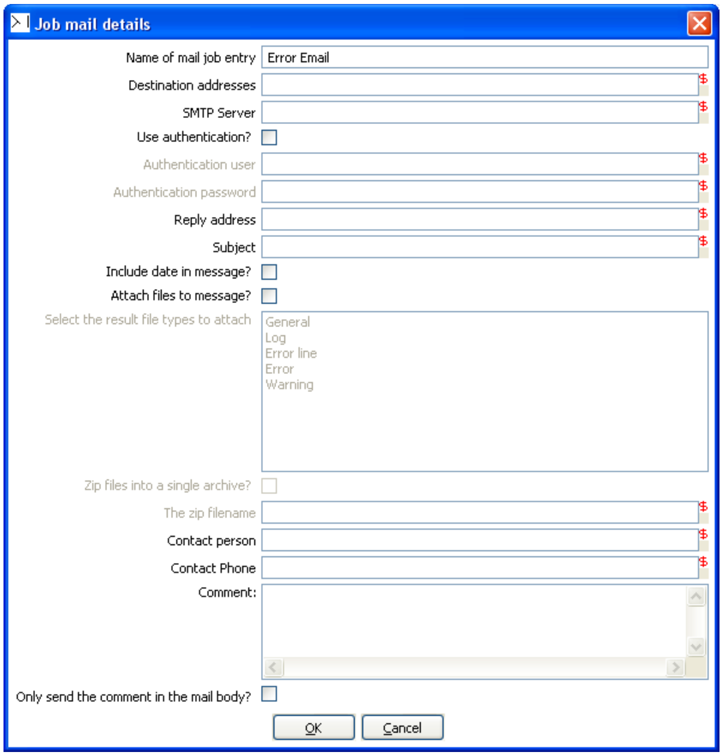
NOTE: *On Windows, scripts are preceded by "CMD.EXE /C" (NT/XP/2000) or "COMMAND.COM /C" (95,98).*

13.2.4.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Filename	The filename of the shell script to start, should include full path else ${user.dir} is used as path.
Working directory	The directory that will be used as working directory for the shell script. The working directory only becomes active when the shell script starts so "Filename" should still include the full path to the script. When the field is left empty or the working directory is invalid ${user.dir} will be used as working directory.
Specify log file	Check this if you want to specify a separate logging file for the execution of this shell script.
Name of log file	The directory and base name of the log file (for example C:\logs)
Extension of the log file	The filename extension (for example: log or txt)
Include date in filename	Adds the system date to the filename. (_20051231)
Include time in filename	Adds the system time to the filename. (_235959)
Logging level	Specifies the logging level for the execution of the shell. See also the logging window in 15. Logging
Copy previous results to arguments	The results from a previous transformation can be sent to the shell script using the "Copy rows to result" step. (as arguments)
Arguments	Specify the strings to use as arguments for the shell script.
Execute once for every input row	This implements looping. If the previous job entry returns a set of result rows, you can have this shell script executed once for every row found. One row is passed to this script at every execution in combination with the copy previous result to arguments. The values of the corresponding result row can then be found on command line argument $1, $2, ... (%1, %2, %3, ... on Windows)

13.2.5. Mail

13.2.5.1. Screenshot



Job mail details

Name of mail job entry: Error Email

Destination addresses:

SMTP Server:

Use authentication? ☐

Authentication user:

Authentication password:

Reply address:

Subject:

Include date in message? ☐

Attach files to message? ☐

Select the result file types to attach: General, Log, Error line, Error, Warning

Zip files into a single archive? ☐

The zip filename:

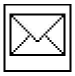
Contact person:

Contact Phone:

Comment:

Only send the comment in the mail body? ☐

13.2.5.2. Icon



13.2.5.3. General description

You can use the Mail job entry to send an e-Mail.

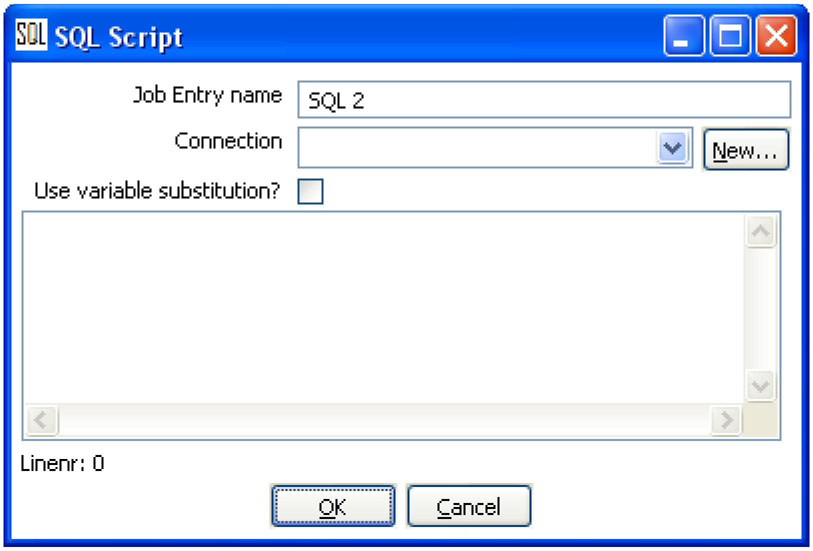
13.2.5.4. Options

Option Description	
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Destination address	The destination for the e-Mail
Use authentication	Check this if your SMTP server requires you to authenticate yourself.
Authentication user	The user name to authenticate with
Authentication password	The password to authenticate with.
SMTP server	The mail server to which the mail has to be sent.
Reply address	The reply address for this e-Mail
Subject	The subject of the e-Mail
Include date in message	Check this if you want to include the date in the e-Mail
Contact person	The name of the contact person to be placed in the e-Mail
Contact phone	The contact telephone number to be placed in the e-Mail

Option Description	
Comment	Additional comment to be placed in the e-Mail
Attach files to message	Check this if you want to attach files to this message.
Select the result files types to attach.	When a transformation (or job) processes files (text, excel, dbf, etc) an entry is being added to the list of files in the result of that transformation or job. Specify the types of result files you want to add.
Zip files into a single archive	Check this if you want to zip all selected files into a single archive (recommended!) The zip filename Specify the name of the zip file that will be placed into the e-mail. **NOTE:** *All text fields can be specified using (environment and Kettle) variables, possibly set in a previous transformation using the Set Variable step.*

13.2.6. SQL

13.2.6.1. Screenshot



Job Entry name: SQL 2

Connection:

Use variable substitution? □

Linenr: 0

13.2.6.2. Icon

13.2.6.3. General description

You can use the SQL job entry to execute an SQL script.

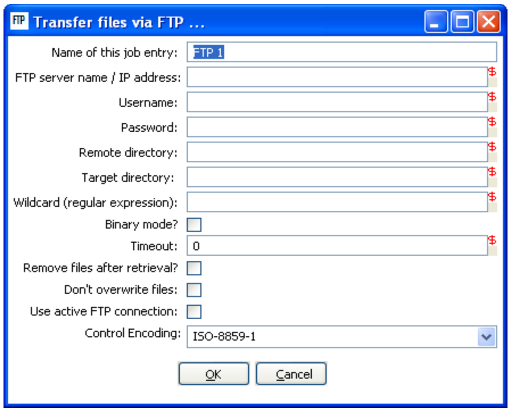
This means a number of SQL statements separated by ;

13.2.6.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Connection	The database connection to use
Use variable substitution?	Enables kettle variables to be used in the SQL Script
SQL script	The SQL script to execute

13.2.7. FTP

13.2.7.1. Screenshot



Transfer files via FTP ...

Name of this job entry: FTP 1

FTP server name / IP address:

Username:

Password:

Remote directory:

Target directory:

Wildcard (regular expression):

Binary mode? □

Timeout: 0

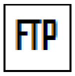
Remove files after retrieval? □

Don't overwrite files: □

Use active FTP connection: □

Control Encoding: ISO-8859-1

13.2.7.2. Icon



13.2.7.3. General description

You can use the FTP job entry to get one or more files from an FTP server.

13.2.7.4. Options

Option	Description
Job entry name	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
FTP server name	The name of the server or the IP address
User name	The user name to log into the FTP server
Password	The password to log into the FTP server
Remote directory	The remote directory on the FTP server from which we get the files
Target directory	The directory on the machine on which Kettle runs in which you want to place the transferred files

Wildcard	Specify a regular expression here if you want to select multiple files.
	For example:
	`.*txt$              : get all text files`
	`A.*[0-9]\.txt       : files tarting with A`
	`                       ending with a number and .txt`

Use binary mode?	Check this if the files need to be transferred in binary mode.
Timeout	The FTP server timeout in seconds.
Remove files after retrieval?	Remove the files on the FTP server, but only after all selected files have been successfully transferred.
Don't overwrite files	Skip a file when a file with identical name already exists in the target directory.
Use active FTP connection	Check this to use active mode FTP instead of the passive mode (default).
Control Encoding	The encoding to use for the ftp control instructions, the encoding matters e.g. for the ftp'ing of filenames when they contain special characters. For Western Europe

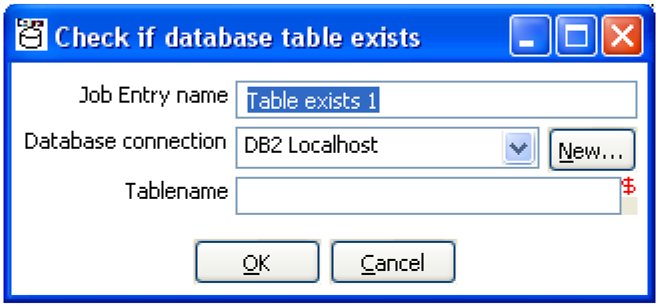
	and the USA "ISO-8859-1" should suffice. You can enter any encoding that is valid on your server.

13.2.7.5. Notes

Some ftp servers don't allow files to be ftp'ed when they contain certain characters (spaces e.g.). So when choosing filenames for files to be ftp'ed, be sure to check upfront whether your particular ftp server is able to process your kind of filenames.

13.2.8. Table Exists

13.2.8.1. Screenshot



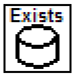
Check if database table exists

Job Entry name: Table exists 1

Database connection: DB2 Localhost

Tablename:

13.2.8.2. Icon



13.2.8.3. General description

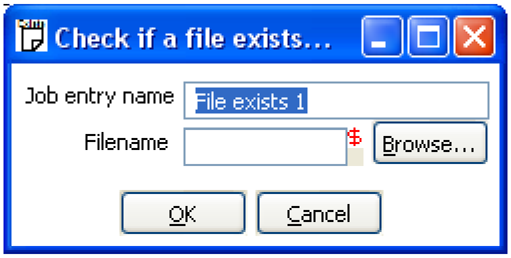
You can use the Table Exists job entry to verify if a certain table exists on a database.

13.2.8.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Database connection	The database connection to use
Tablename	The name of the database table to check

13.2.9. File Exists

13.2.9.1. Screenshot

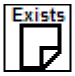


Check if a file exists...

Job entry name: File exists 1

Filename:

13.2.9.2. Icon



13.2.9.3. General description

You can use the File Exists job entry to verify if a certain file exists on the server on which Kettle runs.

13.2.9.4. Options

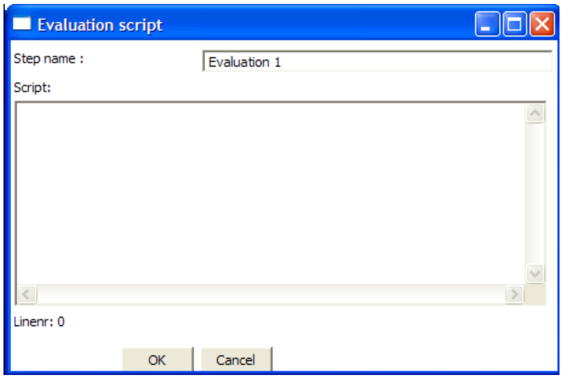
Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Filename	The name and path of the file to check for

13.2.9.5. Extra

- Variable: select the variable to use as filename
- Browse: look for the file on the file system

13.2.10. Evaluation

13.2.10.1. Screenshot

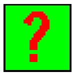


Step name : Evaluation 1

Script:

Linenr: 0

13.2.10.2. Icon



13.2.10.3. General description

You can use the Evaluation job entry to calculate a boolean variable. This variable can then be used to determine which next step will be executed.

The following variables are available to the user:

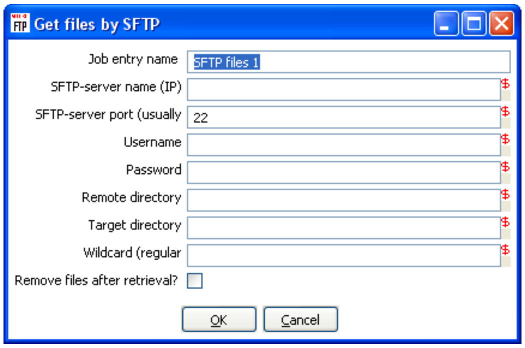
- errors : number of errors in the previous job entry (long)
- lines_input : number of rows read from database or file (long)
- lines_output : number of rows written to database or file (long)
- lines_updated : number of rows updated in a database table (long)
- lines_read : number of rows read from a previous transformation step (long)
- lines_written : number of rows written to a next transformation step (long)
- files_retrieved : number of files retrieved from an FTP server (long)
- exit_status : the exit status of a shell script (integer)
- nr (integer) : the job entry number. Increments at every next job entry.
- is_windows : true if Kettle runs on MS Windows (boolean)

13.2.10.4. Options

Option Description	
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Script	The Javascript that needs to be run to evaluate. The last statement in the script needs to result in a boolean

13.2.11. SFTP

13.2.11.1. Screenshot



Get files by SFTP

Job entry name: SFTP files 1
SFTP-server name (IP):
SFTP-server port (usually: 22
Username:
Password:
Remote directory:
Target directory:
Wildcard (regular:
Remove files after retrieval? □

13.2.11.2. Icon

13.2.11.3. General description

You can use the SFTP job entry to get one or more files from an FTP server using the Secure FTP protocol.

13.2.11.4. Options

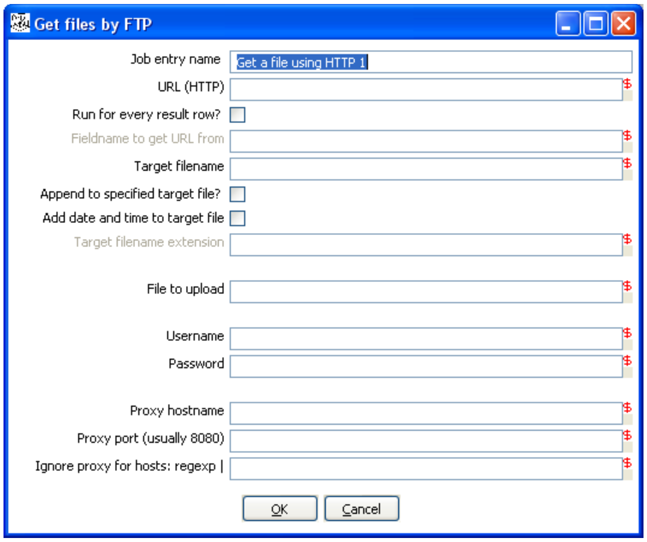
Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
SFTP-server name (IP)	The name of the SFTP server or the IP address
SFTP port	The TCP port to use. This is usually 22
User name	The user name to log into the SFTP server
Password	The password to log into the SFTP server
Remote directory	The remote directory on the SFTP server from which we get the files
Target directory	The directory on the machine on which Kettle runs in which you want to place the transferred files

Wildcard	Specify a regular expression here if you want to select multiple files.
	For example:
	`.*txt$ : get all text files`
	`A.*[0-9]\.txt : files tarting with A ending with a number and .txt`

Remove files after retrieval?	Remove the files after they have been successfully transferred.

13.2.12. HTTP

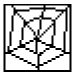
13.2.12.1. Screenshot


Get files by FTP
Run for every result row? ☐
Fieldname to get URL from
Target filename
Append to specified target file? ☐
Add date and time to target file ☐
Target filename extension
File to upload
Username
Password
Proxy hostname
Proxy port (usually 8080)
Ignore proxy for hosts: regexp |
OK
Cancel

Job entry name: Get a file using HTTP 1

URL (HTTP):

13.2.12.2. Icon



13.2.12.3. General description

You can use the HTTP job entry to get a file from a web server using the HTTP protocol.

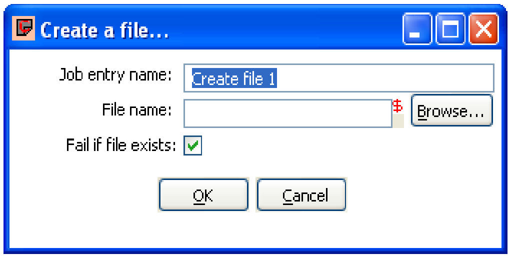
13.2.12.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
URL (HTTP)	The URL to use (for example: http://kettle.pentaho.org)
Run for every result row	Check this if you want to run this job entry for every row that was generated by a previous transformation. Use the "Copy rows to result"
Fieldname to get URL from	The fieldname in the result rows to get the URL from
Target filename The target filename	
Append to target file	Append to the target file if it already exists
Add date and time to target filename	Check this if you want to add date and time yyyMMdd_HHmmss to the target filename.
Target filename extension	Specify the target filename extension in case you're adding a date and time to the filename
Username	The username to authenticate with. For Windows Domains, put the Domain in from of the user like this DOMAIN\Username
Password	The password to authenticate with.
Proxy server	The HTTP proxy server name or IP address

Option	Description
Proxy port	The HTTP proxy port to use (usually 8080)
Ignore proxy for hosts	Specify a regular expression matching the hosts you want to ignore, | separated. For example 127\.0\..*

13.2.13. Create a file

13.2.13.1. Screenshot



Create a file...

Job entry name: Create file 1

File name:

Fail if file exists: ☑

13.2.13.2. Icon

13.2.13.3. General description

You can use the Create a file job entry to create an empty file. This is useful for creating "trigger" files from within jobs.

13.2.13.4. Options

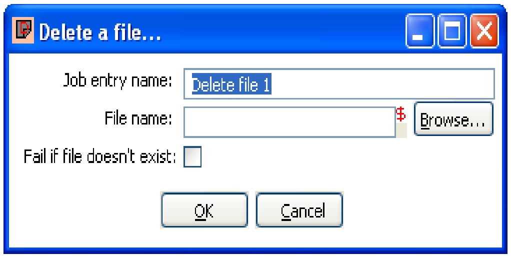
Option	Description
Job entry name	The name of the job entry. This name has to be unique in a single job.
File name	The name and path of the file to create an empty file.
Fail if file exists	The job entry will follow the failure outgoing hop when the file to be created already exists (empty or not) and this option is switched on. The default is on.

13.2.13.5. Extra

- Variable: select the variable to use as filename
- **Browse: look for the file on the file system**

13.2.14. Delete a file

13.2.14.1. Screenshot



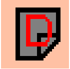
Delete a file...

Job entry name: Delete file 1

File name:

Fail if file doesn't exist: □

13.2.14.2. Icon



13.2.14.3. General description

You can use the Delete a file job entry to delete a file (empty or not).

13.2.14.4. Options

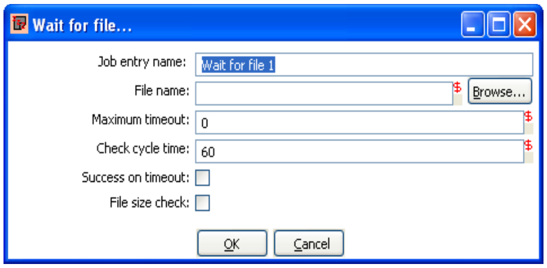
Option	Description
Job entry name	The name of the job entry. This name has to be unique in a single job.
File name	The name and path of the file to delete.
Fail if file doesn't exist	The job entry will follow the failure outgoing hop when the file to be deleted does not exist anymore and this option is switched on. The default is off.

13.2.14.5. Extra

- Variable: select the variable to use as filename
- **Browse: look for the file on the file system**

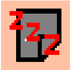
13.2.15. Wait for a file

13.2.15.1. Screenshot



Job entry name: Wait for file 1
File name:
Maximum timeout: 0
Check cycle time: 60
Success on timeout: □
File size check: □

13.2.15.2. Icon



13.2.15.3. General description

You can use the Wait for file job entry to wait for a file. This job entry will sleep and periodically check whether the specified file exists after which the flow will continue. The job entry can either wait indefinitely for the file or it can timeout after a certain time.

13.2.15.4. Options

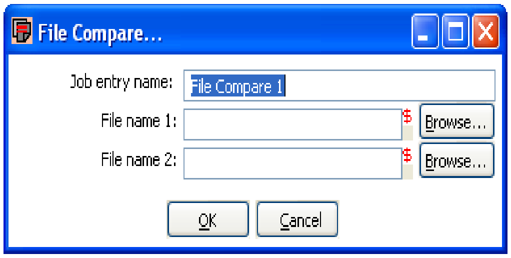
Option	Description
Job entry name	The name of the job entry. This name has to be unique in a single job.
File name	The name and path of the file to wait for.
Maximum timeout	The maximum timeout in number of seconds, or 0 to wait indefinitely. This is the number seconds after which the flow will continue even if the file wasn't created. When the timeout is reached the "Success on timeout" option will determine whether the outgoing success or failure hop will be followed.
Check cycle time	The time in seconds between checking for the file. The file will be checked for in the start of the execution and then every "check cycle time" seconds until the maximum timeout is reached. A job can only be stopped every "check cycle time" as else the job entry step will be sleeping. A check cycle time of 30 or 60 seconds seems to be a good trade-off between the period until the file is detected and the required CPU usage.
Success on timeout	What to do when the "Maximum timeout" has been reached, when switched on the success outgoing hop will be followed if the file is not detected after timeout.
File size check	When this is switched on the job entry will after detecting the specified file, only continue if the file size hasn't changed the last check "cycle time seconds". This is useful e.g. if a file is created in place (although it's recommended to generate a file elsewhere and then move it in place).

13.2.15.5. 12.2.15.5. Extra

- Variable: select the variable to use as filename
- **Browse: look for the file on the file system**

13.2.16. File compare

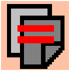
13.2.16.1. Screenshot



File Compare...

Job entry name: File Compare 1

13.2.16.2. Icon



13.2.16.3. General description

You can use the File compare job entry to compare the contents of 2 files and control the flow of the job by it. When the contents of the files are the same the success outgoing hop will be followed, else the failure hop will be followed.

13.2.16.4. Options

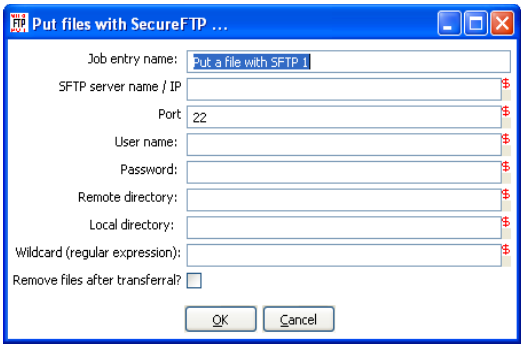
Option	Description
Job entry name	The name of the job entry. This name has to be unique in a single job.
File name 1	The name and path of the file of the first file to compare.
File name 2	The name and path of the file of the second file to compare.

13.2.16.5. Extra

- Variable: select the variable to use as filenames
- **Browse: look for the files on the file system**

13.2.17. Put files with SecureFTP

13.2.17.1. Screenshot



Put files with SecureFTP ...

Job entry name: Put a file with SFTP 1

SFTP server name / IP

Port 22

User name:

Password:

Remote directory:

Local directory:

Wildcard (regular expression):

Remove files after transferral? □

13.2.17.2. Icon

13.2.17.3. General description

You can use the Put files with SFTP job entry to put one or more files from an FTP server using the Secure FTP protocol.

13.2.17.4. Options

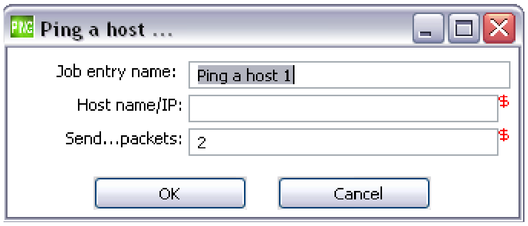
Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
SFTP-server name (IP)	The name of the SFTP server or the IP address
SFTP port	The TCP port to use. This is usually 22
User name	The user name to log into the SFTP server
Password	The password to log into the SFTP server
Remote directory	The remote directory on the SFTP server to which we put the files
Local directory	The directory on the machine on which Kettle runs from which you want to ftp the files from

Wildcard	Specify a regular expression here if you want to select multiple files.
	For example:
	`.*txt$            : get all text files`
	`A.*[0-9]\.txt      : files tarting with A ending with a number and .txt`

Remove files after transferral?	Remove the files after they have been successfully transferred.

13.2.18. Ping a host

13.2.18.1. Screenshot



Job entry name: Ping a host 1

Host name/IP:

Send...packets: 2

13.2.18.2. Icon

13.2.18.3. General description

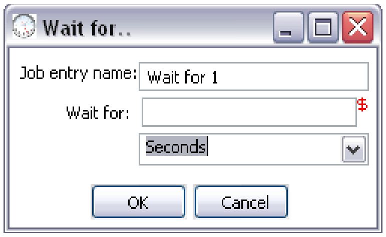
You can use the Ping a host job entry to ping a host using the ICMP protocol.

13.2.18.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Host name/IP	The name or the IP address of the host to ping
Send...packets	The number of packets to send (by default 2)

13.2.19. Wait for

13.2.19.1. Screenshot



Job entry name: Wait for 1

Wait for:

Seconds

13.2.19.2. Icon

13.2.19.3. General description

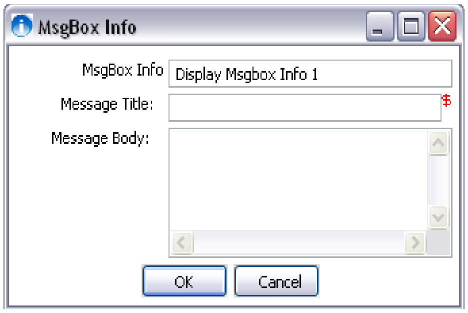
You can use the Wait for to wait a delay before running the next job entry.

13.2.19.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Wait for	The delay to wait
Unit time	Specify the unit time (second, minute and hour)

13.2.20. Display Msgbox info

13.2.20.1. Screenshot



MsgBox Info

MsgBox Info: Display Msgbox Info 1

Message Title:

Message Body:

13.2.20.2. Icon

13.2.20.3. General description

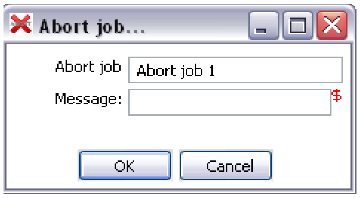
This job entry allow you to display a message box in job. You can easily see where you are in the process. This entry is only available via GUI.

13.2.20.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Message title	The title of the message
Message body	The message to display

13.2.21. Abort job

13.2.21.1. Screenshot



Abort job...

Abort job: Abort job 1

Message:

13.2.21.2. Icon

13.2.21.3. General description

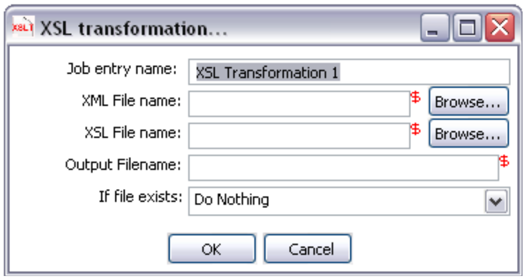
Use this job entry if you want to abort a job.

13.2.21.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Message	Message to add in log when aborting

13.2.22. XSL transformation

13.2.22.1. Screenshot



Job entry name: XSL Transformation 1

XML File name:

XSL File name:

Output Filename:

If file exists: Do Nothing

13.2.22.2. Icon

13.2.22.3. General description

XSL transformation job entry is designed to transform (by applying XSL document) XML documents into other documents (XML or other format, such as HTML or plain text).

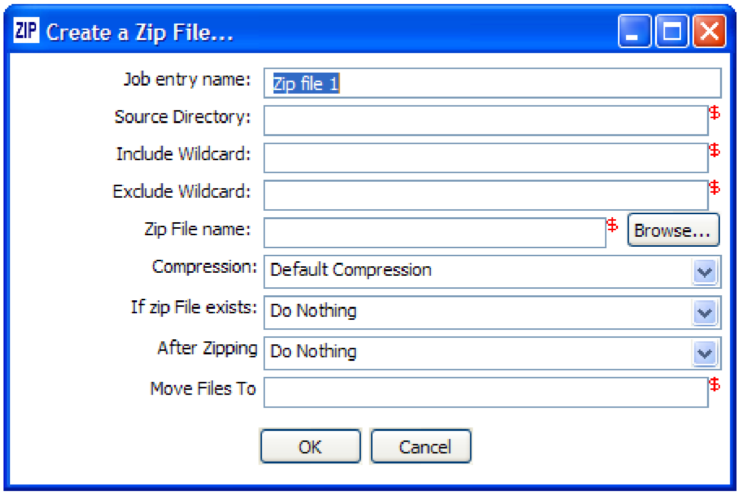
The original document is not changed; rather, a new document is created based on the content of the XML file.

13.2.22.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
XML File name	The full name of the source XML file
XSL File name	The full name of the XSL file
Output File name	The full name of the created document (result of XSL transformation)
If file exists	Define the behavior when an output file with the same name exists Options : – Create new with unique name : a new output file will be created – Do nothing : nothing will be done – Fail : the job will fail

13.2.23. Zip files

13.2.23.1. Screenshot



Create a Zip File...

Job entry name: Zip file 1

Source Directory:

Include Wildcard:

Exclude Wildcard:

Zip File name:

Compression: Default Compression

If zip File exists: Do Nothing

After Zipping: Do Nothing

Move Files To

13.2.23.2. Icon

13.2.23.3. General description

This step creates a standard ZIP archive using the options you specify in the dialog.

13.2.23.4. Options

Option	Description
Name of the job entry	The name of the job entry. This name has to be unique in a single job. A job entry can be placed several times on the canvas, however it will be the same job entry.
Source directory	The source directory of the files to be zipped
Include wildcard	The wildcard (regular expression) of the files to include in the zip archive
Exclude wildcard	The wildcard (regular expression) of the files to exclude from the zip archive
Zip file name	The full name of the destination archive
Compression	The compression level to be used (Default, Best Compression, Best speed)
If zip file exists	The action to take when there already is a file at the target destination.
After zipping	The action to take after zipping
Move files to	The target directory to move the source files to after zipping

14. Graphical View

14.1. Description

The Graphical View tab contains the canvas on which transformations and jobs are drawn. There will be a separate tab for each job and/or transformation you currently have open.

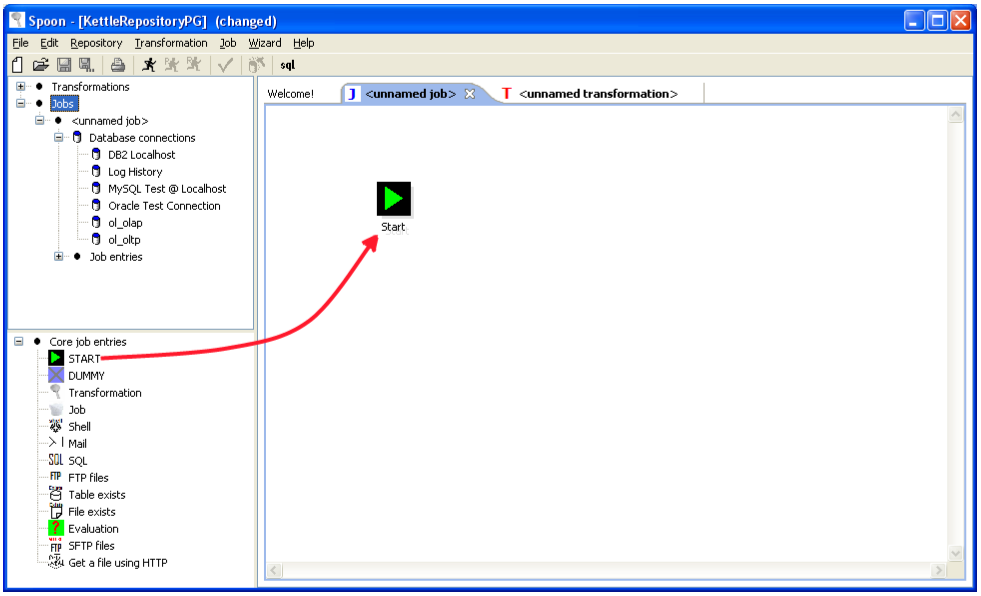
J	T
Job Tab	*Transformation Tab*

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

14.2. Adding steps or job entries

14.2.1. Create steps by drag and drop

Adding steps to a transformation (or a job entry to a job) 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.

14.2.2. 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.

14.2.3. Create steps on the place where you like them

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

14.3. Hiding a step

If you right click on a step or job entry 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.

14.4. Transformation Step options (right click popup menu)

14.4.1. Edit step

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

14.4.2. Edit description

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

14.4.3. Copy data / distribute data

See 7.3. Distribute or copy?

14.4.4. Copies

See 7.2. Launching several copies of a step

14.4.5. Duplicate step

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

14.4.6. Delete step

This will permanently remove the step from the transformation.

14.4.7. Show input fields

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

14.4.8. Show output fields

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

14.5. Job entry options (right click popup menu)

14.5.1. Open Transformation/Job

This opens a Spoon or Chef window displaying the selected transformation or job.

14.5.2. Edit job entry

This opens the dialog for the job entry allow you to change the settings

14.5.3. Edit job entry description

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

14.5.4. Create shadow copy of job entry

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

14.5.5. Copy selected entries to clipboard (CTRL-C)

Copies the XML describing the selected job entries to the clipboard.

14.5.6. Align / distribute

This option allows you to keep the graph clean by aligning job entries with each other.

14.5.7. Detach entry

Unlinks this job entry from the hops that connect it to other steps.

14.5.8. Delete all copies of this entry.

Delete all copies of this job entry, not just this one!

14.6. 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 SHIFT key at the same time.

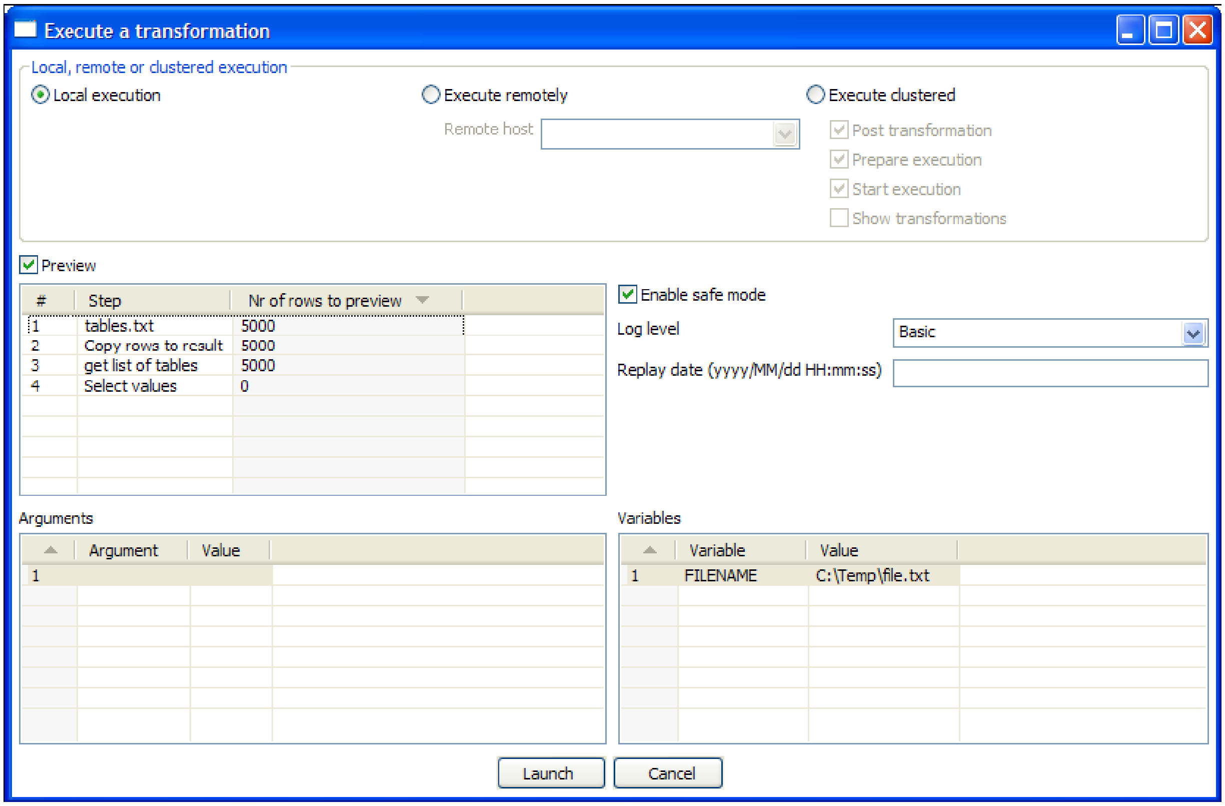
For a more complete explanation regarding hops, please check out 7. Hops.

Running a Transformation

14.7. Running a Transformation Overview

When you are finished modifying your transformation, you can run it by clicking on the run button from the main menu, toolbar or by pressing F9.

14.8. Screenshot



Execute a transformation

Local, remote or clustered execution

● Local execution ○ Execute remotely ○ Execute clustered

Remote host

☑ Post transformation

☑ Prepare execution

☑ Start execution

□ Show transformations

☑ Preview

#	Step	Nr of rows to preview
1	tables.txt	5000
2	Copy rows to result	5000
3	get list of tables	5000
4	Select values	0

☑ Enable safe mode

Log level: Basic

Replay date (yyyy/MM/dd HH:mm:ss)

Arguments

	Argument	Value
1		

Variables

	Variable	Value
1	FILENAME	C:\Temp\file.txt

14.9. Execution Options

14.9.1. Where to Execute

There are three options for deciding where you want your transformation to be executed:

- **Local Execution**: the transformation or job will be run on the machine you are currently using
- **Execute remotely**: allows you to specify a remote server where you want the execution to take place. This feature requires that you have Pentaho Data Integration installed on a remote machine

and running the Carte service. See the 14.4.3 Configuring a remote or slave server for more details on setting up remote and slave servers.

- **Execute clustered**: Allows you to execute the job or transformation in a clustered environment. See the section on Clustering for more details on how to execute a job or transformation in a clustered environment.

14.9.2. Preview

Preview allows you to specify a number of rows for each step that you would like to preview.

14.9.3. Enable safe mode

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

14.9.4. Log Level

This allows you to specify the level of detail you want to capture in the log. For detailed descriptions of the log level types see 15. Logging.

14.9.5. Replay Date

This will set the replay date for when you want to replay the transformation. It will pick up information in the text file input or Excel input steps to skip rows already processed on the replay date.

14.9.6. Arguments

This grid allows you to set the value of arguments to be used when running the transformation.

14.9.7. Variables

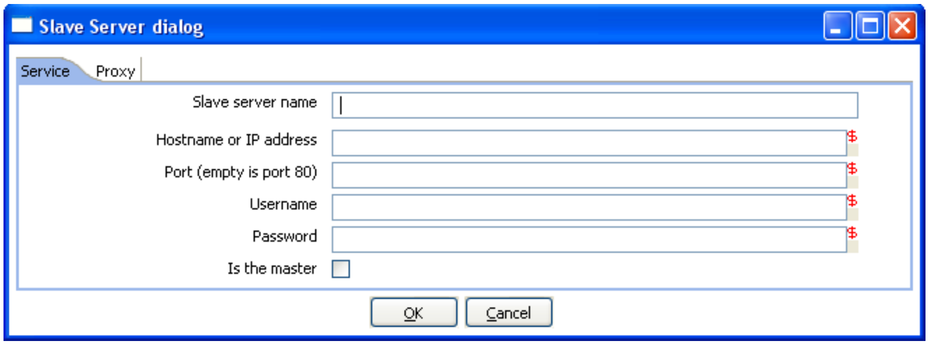
This grid allows you to set the value of variables to be used when running the transformation.

14.10. Setting up Remote and Slave Servers

14.10.1. Overview

Slave servers allow you to execute a transformation on a remote server. Setting up a slave server requires having a small web-server running on your remote machine called “Carte” that will accept input from either Spoon (remote & clustered execution) or from the Transformation job entry (clustered execution).

14.10.2. Screen shot



14.10.3. Configuring a remote or slave server

Install Pentaho Data Integration on server you want to use to remotely execute transformations (for more information, see 2.2. Installation). The installation includes a small web server called Carte used to support remote requests. Start the Carte server by running Carte.bat (Windows) or carte.sh from the root of your Pentaho Data Integration installation.

Next, you need to point your master server to each of the slave server. To do this, double click on 'Slave server' node in the tree control on the left, or by right-clicking on 'Slave Server' and selecting the New Slave Server option.

14.10.3.1. Service Tab

- **Slave server name**: The friendly name of the server you wish to use as a slave
- **Hostname or IP address**: The address of the machine to be used as a slave
- **Port**: Defines the port you wish to use for communicating with the remote server
- **Username**: Enter the username credential for accessing the remote server
- **Password**: Enter the password credential for accessing the remote server
- **Is the master**: This setting tells Pentaho Data Integration that this server will act as the master server in any clustered executions of the transformation

  Note: *when executing a transformation in a clustered environment, you should have 1 server setup as the master and all remaining servers in the cluster as slaves.*

14.10.3.2. Proxy Tab

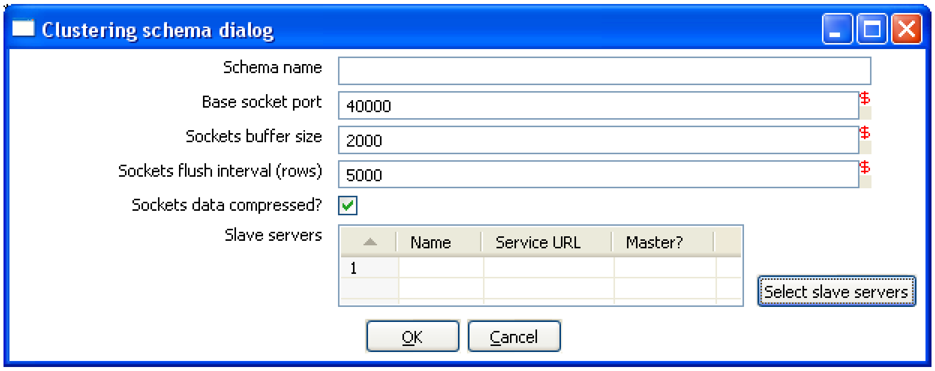
- **Proxy server hostname**: Sets the hostname for the Proxy server you are connecting through
- **The proxy server port**: Sets the port number used in communication with the proxy
- **Ignore proxy for hosts: regexp|separated**: Specify the server(s) for which the proxy should not be active. This option supports specifying multiple servers using regular expressions. You can also add multiple servers and expressions separated by the '|' character.

14.11. Clustering

14.11.1. Overview

Clustering allows transformations and transformation steps to be executed in parallel on more than one server. The clustering schema defines which slave servers you want to assign to the cluster and a variety of clustered execution options.

14.11.2. Screen shot



Clustering schema dialog

Schema name:

Base socket port: 40000

Sockets buffer size: 2000

Sockets flush interval (rows): 5000

Sockets data compressed? ☑

Slave servers:

	Name	Service URL	Master?
1			

14.11.3. Creating a cluster schema

Begin by double-clicking on the 'Kettle cluster schemas' node in the tree on the left or right-clicking on that node and selecting 'New clustering schema'.

Clustering schema options:

- **Schema name**: The name of the clustering schema
- **Base socket port**: Here you can specify the port from which to start numbering ports for the slave servers. Each additional clustered step executing on a slave server will consume an additional port.
  Note: *Make sure no other networking protocols are in the same range to avoid networking problems.*
- **Sockets buffer size**: The internal buffer size to use.
- **Sockets flush interval (rows)**: The amount of rows after which the internal buffer is sent completely over the network and emptied.
- **Sockets data compressed**: when this is checked, all data is compressed using the Gzip compression algorithm to minimize network traffic.
- **Slave servers**: This is a list of the servers to be used in the cluster. You should have one master server and any number of slave servers. To add servers to the cluster, click on the 'Select slave servers' button to select from the list of available slave servers. See 14.4. Configuring a remote or slave server for more details on how to create a slave server.

14.11.4. Running transformations using a cluster

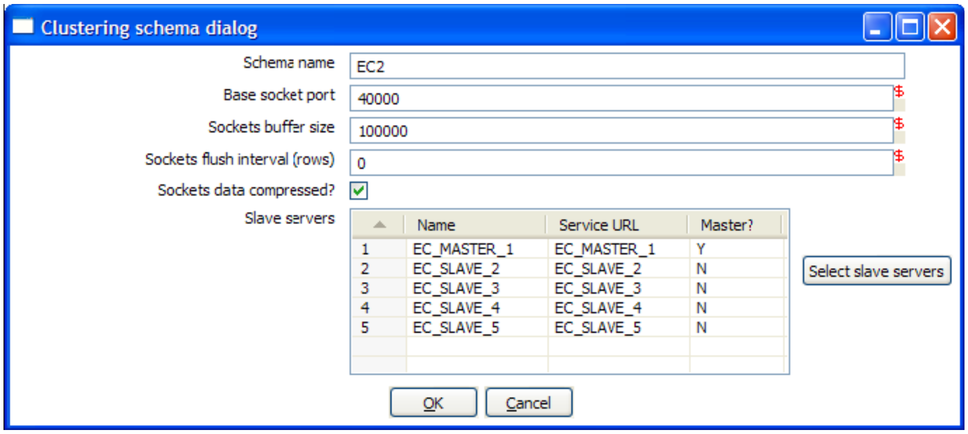
When you chose to run a Transformation, select the 'Execute clustered'. You will have the following options:

- **Post transformation**: Split the transformation and post it to the different master and slave servers.

- **Prepare execution**: This runs the initialization phase of the transformation on the master and slave servers.
- **Start execution**: This starts the actual execution of the master and slave transformations.
- **Show transformations**: Show the generated (converted) transformations that will be executed on the cluster (see 14.5.5. Basic Clustering Example for more information generated transformations).

14.11.5. Basic Clustering Example

Suppose that you have data from a database table and you want to run it through a particularly complex JavaScript program.
For performance reasons you want to execute this program on 5 different hosts.
You create a cluster with one master server and 4 slaves:

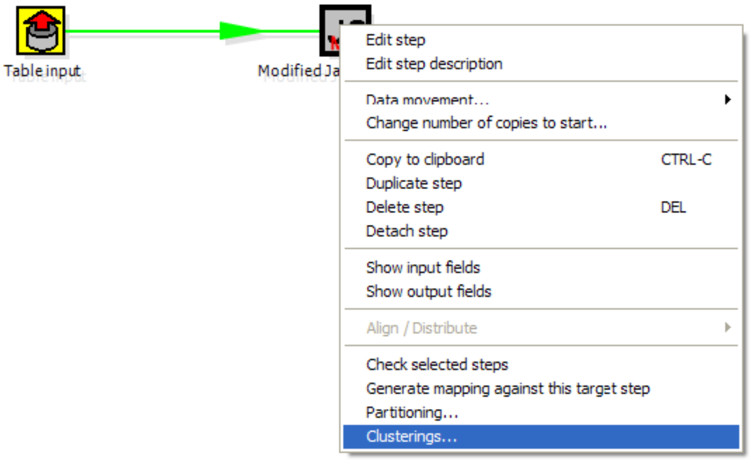


Clustering schema dialog

Schema name: EC2
Base socket port: 40000
Sockets buffer size: 100000
Sockets flush interval (rows): 0
Sockets data compressed? ☑
Slave servers

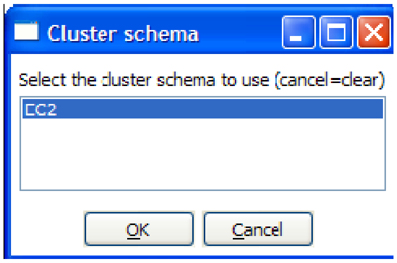
	Name	Service URL	Master?
1	EC_MASTER_1	EC_MASTER_1	Y
2	EC_SLAVE_2	EC_SLAVE_2	N
3	EC_SLAVE_3	EC_SLAVE_3	N
4	EC_SLAVE_4	EC_SLAVE_4	N
5	EC_SLAVE_5	EC_SLAVE_5	N

Then you create the transformation as usual, connecting 2 steps with a hop.



You specify that the script to execute is running on a cluster:

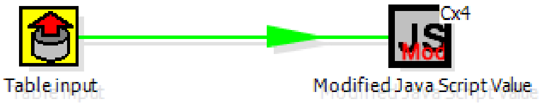
Then select the cluster to use:



Cluster schema

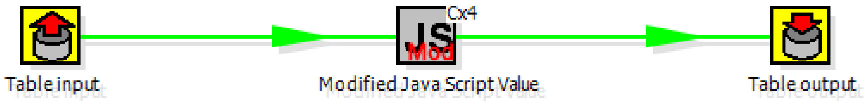
Select the cluster schema to use (cancel=clear)

CC2

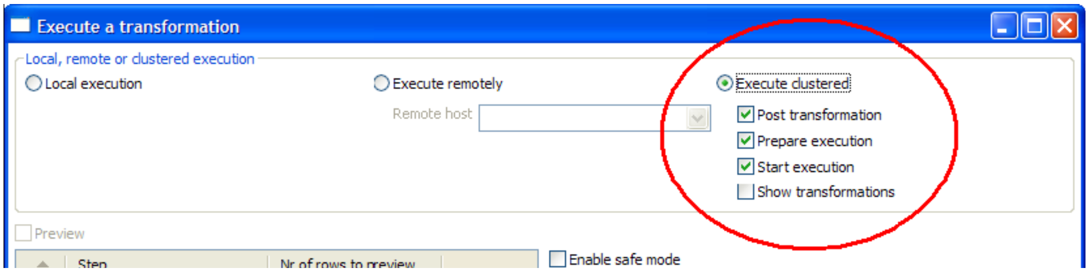


The transformation will then be drawn like this on the graphical view:

The Cx4 indicates that the step will be executed on a cluster.
Suppose we then store the calculated information as a result in a table again:



When we execute this transformation locally, we will see no difference with the usual result you expect from a non-clustered execution. That means that you can use the normal local execution to test the transformation.



Execute a transformation

Local, remote or clustered execution

○ Local execution

○ Execute remotely

Remote host

● Execute clustered

☑ Post transformation

☑ Prepare execution

☑ Start execution

□ Show transformations

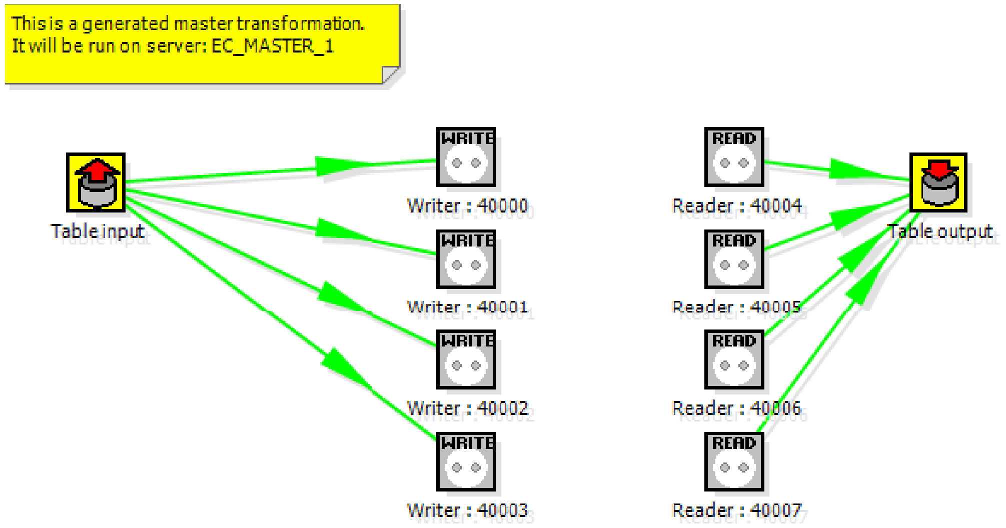
□ Preview

□ Enable safe mode

However when we can execute the transformation in a clustered fashion like this:

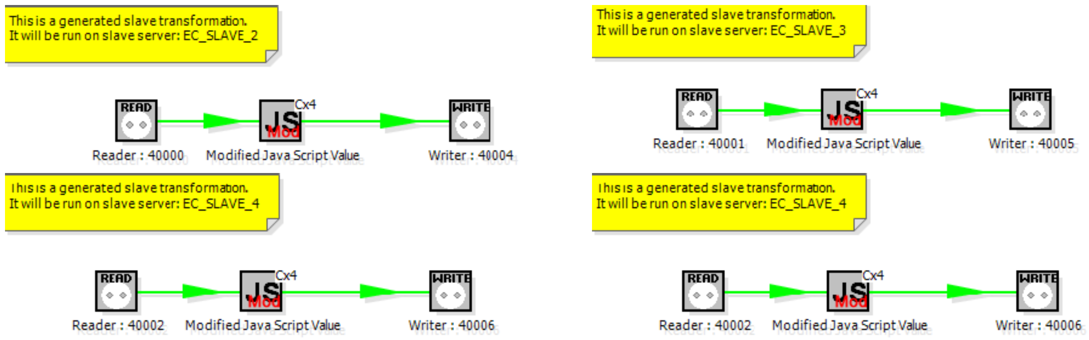
In this case, 5 transformations will be generated for the 5 servers in the cluster.

One master:



This is a generated master transformation.
It will be run on server: EC_MASTER_1

And 4 slaves transformations:



This is a generated slave transformation.
It will be run on slave server: EC_SLAVE_2

This is a generated slave transformation.
It will be run on slave server: EC_SLAVE_3

This is a generated slave transformation.
It will be run on slave server: EC_SLAVE_4

This is a generated slave transformation.
It will be run on slave server: EC_SLAVE_4

As you can see, data will be sent over the TCP/IP sockets using the Socket Writer and Socket Reader steps.

15. Logging

15.1. Logging Description

A log view tab will open automatically each time you run a transformation or job. This shows what's happening when the transformation or job is running.

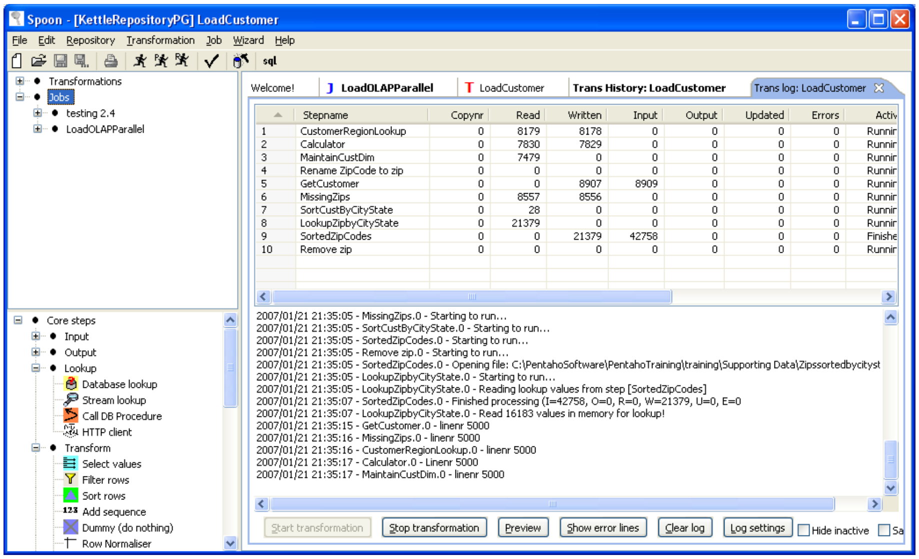
The log tab for transformations will show:

- the (copies of) the steps that are launched as a part of the transformation
- the log as it would be shown if the transformation would be launched by Pan

For jobs, the log view tab will show:

- The details of the completed job entries
- The log as it would be shown if the job would be launched by kitchen

15.2. Screenshots



	Stepname	Copynr	Read	Written	Input	Output	Updated	Errors	Activ
1	CustomerRegionLookup	0	8179	8178	0	0	0	0	Runnir
2	Calculator	0	7830	7829	0	0	0	0	Runnir
3	MaintainCustDim	0	7479	0	0	0	0	0	Runnir
4	Rename ZipCode to zip	0	0	0	0	0	0	0	Runnir
5	GetCustomer	0	0	8907	8909	0	0	0	Runnir
6	MissingZips	0	8557	8556	0	0	0	0	Runnir
7	SortCustByCityState	0	28	0	0	0	0	0	Runnir
8	LookupZipbyCityState	0	21379	0	0	0	0	0	Runnir
9	SortedZipCodes	0	0	21379	42758	0	0	0	Finishe
10	Remove zip	0	0	0	0	0	0	0	Runnir

```
2007/01/21 21:35:05 - MissingZips.0 - Starting to run...
2007/01/21 21:35:05 - SortCustByCityState.0 - Starting to run...
2007/01/21 21:35:05 - SortedZipCodes.0 - Starting to run...
2007/01/21 21:35:05 - Remove zip.0 - Starting to run...
2007/01/21 21:35:05 - SortedZipCodes.0 - Opening file: C:\PentahoSoftware\PentahoTraining\training\Supporting Data\Zipssortedbycityst
2007/01/21 21:35:05 - LookupZipbyCityState.0 - Starting to run...
2007/01/21 21:35:05 - LookupZipbyCityState.0 - Reading lookup values from step [SortedZipCodes]
2007/01/21 21:35:07 - SortedZipCodes.0 - Finished processing (I=42758, O=0, R=0, W=21379, U=0, E=0
2007/01/21 21:35:07 - LookupZipbyCityState.0 - Read 16183 values in memory for lookup!
2007/01/21 21:35:15 - GetCustomer.0 - linenr 5000
2007/01/21 21:35:16 - MissingZips.0 - linenr 5000
2007/01/21 21:35:16 - CustomerRegionLookup.0 - linenr 5000
2007/01/21 21:35:17 - Calculator.0 - Linenr 5000
2007/01/21 21:35:17 - MaintainCustDim.0 - linenr 5000
```

Log Grid

Log Text

Buttons

Transformation Log View Tab

15.3. Log Grid

The log grid is actually a tree that offers a hierarchical view on the execution of a transformation or job.

15.3.1. Transformation Log Grid

- 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.*

15.3.2. Job Log Grid

- The name of the job / job entry
- A comment on the state of the entry execution
- The result (success or failure) of the job entry
- Reason: why was this job entry started?
- The value of the nr variable in the result object (available in evaluation Javascript)
- Log date: logging date, corresponds with the start or end of the job entry.

15.4. 15.4. Buttons

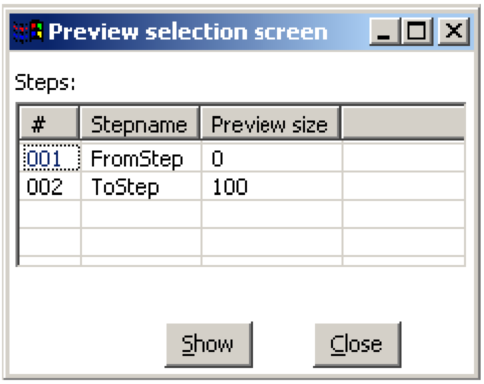
15.4.1. 15.4.1 Transformation Buttons

15.4.1.1. 15.4.1.1. 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.

15.4.1.2. 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:



Steps:

#	Stepname	Preview size
001	FromStep	0
002	ToStep	100

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.

15.4.1.3. Show error lines

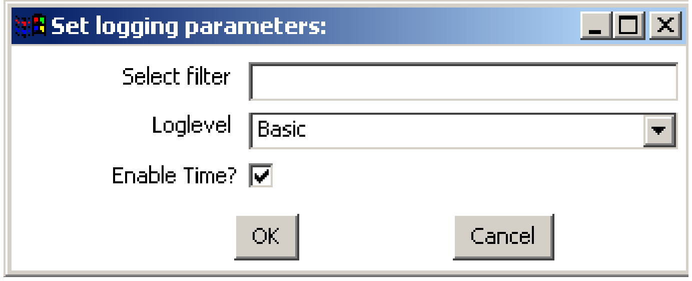
This option will show you all lines of the Log Text that contain the word ERROR (lower- or uppercase). You can then choose to edit the source step of the error.

15.4.1.4. Clear log

This clears the text in the Log Text Window.

15.4.1.5. Log Settings

This is the "Log Settings" dialog:



Set logging parameters:

Select filter

Loglevel: Basic

Enable Time? ☑

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.

15.4.1.6. Hide inactive

Checking this hides all steps that finished processing.

15.4.1.7. Safe mode

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

15.4.2. Job Buttons

15.4.2.1. Start job

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

15.4.2.2. Stop job

This button stops a running job.

15.4.2.3. Refresh log

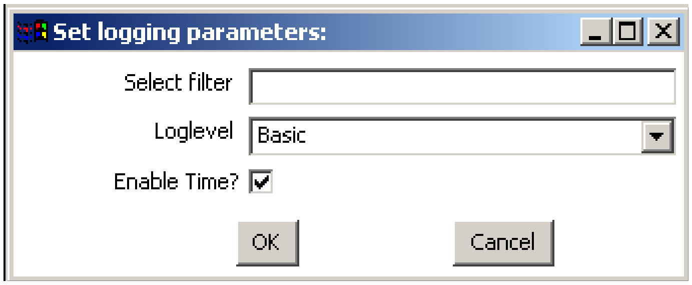
Refreshes the log window.

15.4.2.4. Clear log

This clears the text in the Log Text Window.

15.4.2.5. Log Settings

This is the "Log Settings" dialog:



Set logging parameters:

Select filter

Loglevel: Basic

Enable Time? ☑

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:

Error:	Only show errors
Nothing:	Don't show any output
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.

15.4.2.6. Auto-refresh

If you rather not have the logging window updating all the time you can disable this option. You might want to do this when you're using a remote desktop (VNC, X11) over a slow network connection.

16. Grids

16.1. Description

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

16.2. 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 then there are to keep, simply select the lines you want to keep and use this function. They 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.

16.3. 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.

17. Repository Explorer

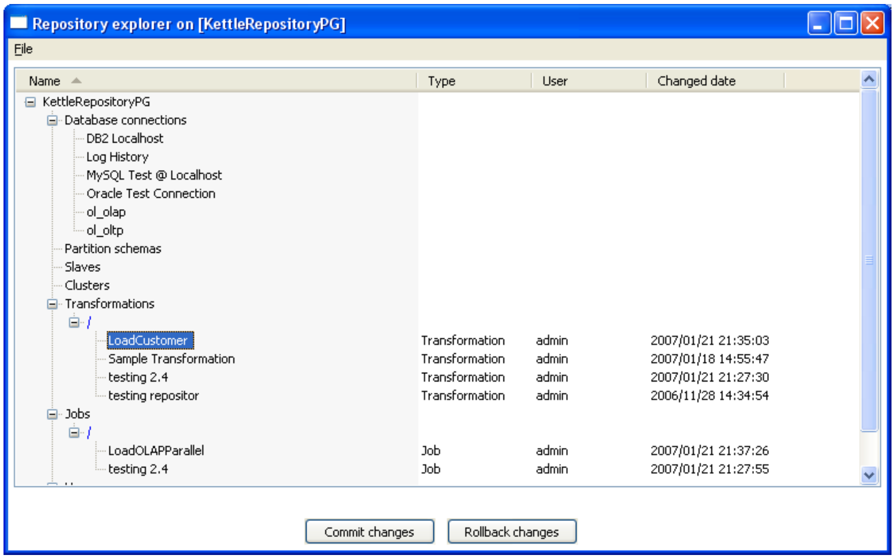
17.1. Description

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 from the repository including:

- Connections
- Partition Schemas
- Slave servers
- Clusters
- Transformations

You can also click on the column header to sort content by name, object type, user or changed date.

17.2. Screenshot



Repository explorer on [KettleRepositoryPG]

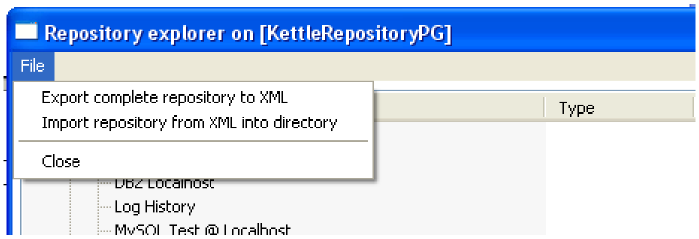
Name	Type	User	Changed date
KettleRepositoryPG			
Database connections			
DB2 Localhost			
Log History			
MySQL Test @ Localhost			
Oracle Test Connection			
ol_olap			
ol_oltp			
Partition schemas			
Slaves			
Clusters			
Transformations			
/			
LoadCustomer	Transformation	admin	2007/01/21 21:35:03
Sample Transformation	Transformation	admin	2007/01/18 14:55:47
testing 2.4	Transformation	admin	2007/01/21 21:27:30
testing repositor	Transformation	admin	2006/11/28 14:34:54
Jobs			
/			
LoadOLAPParallel	Job	admin	2007/01/21 21:37:26
testing 2.4	Job	admin	2007/01/21 21:27:55

17.3. Right click functions

Right clicking on an object in the repository will bring up basic functions such as open, delete and rename objects.

17.4. Backup / Recovery

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



Repository explorer on [KettleRepositoryPG]
File
Export complete repository to XML
Import repository from XML into directory
Close
Type
DB2 Localhost
Log History
MySQL Test @ Localhost

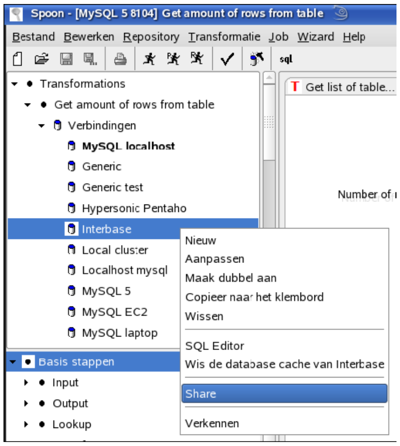
NOTE: *you can restore the objects from a backed up repository anywhere in the target repository directory tree.*

18. Shared objects

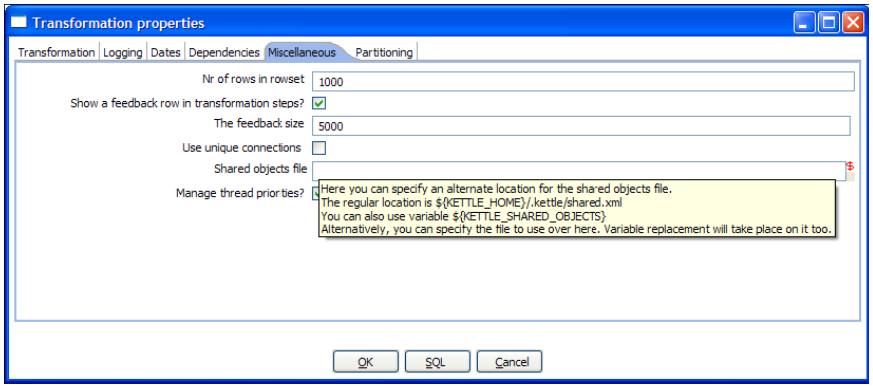
A variety of objects can now be placed in a shared objects file on the local machine. The default location for the shared objects file is $HOME/.kettle/shared.xml. Objects that can be shared using this method include:

- Database connections
- Steps
- Slave servers
- Partition schemas
- Cluster schemas

To share one of these objects, simply right-click on the object in the tree control on the left and choose "share".



The location of the shared objects file is configurable on the "Miscellaneous" tab of the Transformation| Settings dialog:



Nr of rows in rowset: 1000

Show a feedback row in transformation steps? ☑

The feedback size: 5000

Use unique connections ☐

Shared objects file:

Manage thread priorities? ☑

Here you can specify an alternate location for the shared objects file.
The regular location is ${KETTLE_HOME}/.kettle/shared.xml
You can also use variable ${KETTLE_SHARED_OBJECTS}
Alternatively, you can specify the file to use over here. Variable replacement will take place on it too.

19. APPENDIX A

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[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 afee.

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 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 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 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.

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 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!