



Pentaho Data Integration 4.1 User Guide



This document is copyright © 2011 Pentaho Corporation. No part may be reprinted without written permission from Pentaho Corporation. All trademarks are the property of their respective owners.

Help and Support Resources

If you have questions that are not covered in this guide, or if you would like to report errors in the documentation, please contact your Pentaho technical support representative.

Support-related questions should be submitted through the Pentaho Customer Support Portal at <http://support.pentaho.com>.

For information about how to purchase support or enable an additional named support contact, please contact your sales representative, or send an email to sales@pentaho.com.

For information about instructor-led training on the topics covered in this guide, visit <http://www.pentaho.com/training>.

Limits of Liability and Disclaimer of Warranty

The author(s) of this document have used their best efforts in preparing the content and the programs contained in it. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, express or implied, with regard to these programs or the documentation contained in this book.

The author(s) and Pentaho shall not be liable in the event of incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of the programs, associated instructions, and/or claims.

Trademarks

Pentaho (TM) and the Pentaho logo are registered trademarks of Pentaho Corporation. All other trademarks are the property of their respective owners. Trademarked names may appear throughout this document. Rather than list the names and entities that own the trademarks or insert a trademark symbol with each mention of the trademarked name, Pentaho states that it is using the names for editorial purposes only and to the benefit of the trademark owner, with no intention of infringing upon that trademark.

Company Information

Pentaho Corporation
Citadel International, Suite 340
5950 Hazeltine National Drive
Orlando, FL 32822
Phone: +1 407 812-OPEN (6736)
Fax: +1 407 517-4575
<http://www.pentaho.com>

E-mail: communityconnection@pentaho.com

Sales Inquiries: sales@pentaho.com

Documentation Suggestions: documentation@pentaho.com

Sign-up for our newsletter: <http://community.pentaho.com/newsletter/>

Contents

| | |
|--|----|
| Introduction..... | 6 |
| Audience and Assumptions..... | 6 |
| What this Guide Covers..... | 6 |
| Pentaho Data Integration Architecture..... | 7 |
| Pentaho Data Integration Components..... | 7 |
| Starting Pentaho Data Integration..... | 9 |
| Starting the Pentaho Data Integration Servers..... | 9 |
| Starting Spoon..... | 9 |
| Installing and Managing Enterprise Edition License Keys..... | 10 |
| Connecting to the Repository..... | 10 |
| Storing Content Alternatives..... | 10 |
| PDI Interface Perspectives..... | 12 |
| Introducing Perspectives..... | 12 |
| Data Integration (ETL) Perspective..... | 13 |
| VFS File Dialogues in Spoon..... | 14 |
| Modeling Perspective..... | 15 |
| Visualization Perspective..... | 16 |
| Customizing the Spoon Interface..... | 16 |
| Terminology and Basic Concepts..... | 19 |
| Transformations, Steps, and Hops..... | 19 |
| Jobs..... | 20 |
| More About Hops..... | 21 |
| Creating your First Transformation..... | 24 |
| Getting Started..... | 24 |
| Saving Your Transformation..... | 25 |
| Running Your Transformation Locally..... | 26 |
| Building a Job..... | 26 |
| Executing Transformations..... | 29 |
| Setting Up a Slave Server..... | 29 |
| Executing Transformations and Jobs Remotely..... | 31 |
| Creating a Cluster Schema..... | 32 |
| Executing Transformations in a Cluster..... | 33 |
| Impact Analysis..... | 33 |
| Working in the Enterprise Repository..... | 34 |
| Adding an Enterprise Repository..... | 34 |
| Editing Enterprise Repository Details..... | 34 |
| Deleting an Enterprise or Kettle Database Repository..... | 34 |
| Managing Content in the Enterprise Repository..... | 35 |
| Setting Folder-Level Permissions..... | 36 |
| Working with Version Control..... | 37 |
| Examining Version History..... | 38 |
| Restoring a Previously Saved Version of a Job or Transformation..... | 38 |
| Reusing Transformation Flows with Mapping Steps..... | 39 |
| Arguments, Parameters, and Variables..... | 41 |
| Arguments..... | 41 |
| Parameters..... | 41 |
| Configuring SFTP VFS..... | 41 |
| Variables..... | 42 |
| Variable Scope..... | 42 |
| Internal Variables..... | 43 |
| Rapid Analysis Schema Prototyping..... | 44 |
| Creating a Prototype Schema With a Non-PDI Data Source..... | 44 |
| Creating a Prototype Schema With a PDI Data Source..... | 44 |
| Testing With Pentaho Analyzer and Report Wizard..... | 45 |

| | |
|--|-----------|
| Prototypes in Production..... | 45 |
| Managing Connections..... | 46 |
| Adding a JDBC Driver..... | 46 |
| Defining Database Connections..... | 47 |
| Working with JNDI Connections..... | 49 |
| Working with JNDI Connections in Carte and Spoon..... | 49 |
| Database-Specific Options..... | 49 |
| Adding Database-Specific Options..... | 49 |
| Database Connections Advanced Configurations..... | 49 |
| Connection Pooling..... | 50 |
| Clustering..... | 51 |
| Editing, Duplicating, Copying, and Deleting Connections..... | 51 |
| Using the SQL Editor..... | 52 |
| Using the Database Explorer..... | 53 |
| Unsupported Databases..... | 54 |
| Performance Monitoring and Logging..... | 55 |
| Monitoring Step Performance..... | 55 |
| Using Performance Graphs..... | 55 |
| Logging Steps..... | 56 |
| Logging Transformations..... | 57 |
| Pentaho Data Integration Performance Tuning Tips..... | 59 |
| Working With Hadoop..... | 62 |
| Hadoop Job Process Flow..... | 62 |
| Hadoop Transformation Process Flow..... | 64 |
| Hadoop to PDI Data Type Conversion..... | 65 |
| Interacting With Web Services..... | 66 |
| Scheduling and Scripting PDI Content..... | 67 |
| Scheduling Transformations and Jobs From Spoon..... | 67 |
| Command-Line Scripting Through Pan and Kitchen..... | 68 |
| Pan Options and Syntax..... | 68 |
| Kitchen Options and Syntax..... | 69 |
| Transformation Step Reference..... | 72 |
| CSV File Input..... | 73 |
| Excel File Input..... | 75 |
| Excel File Output..... | 77 |
| Fixed File Input..... | 79 |
| Generate Rows..... | 80 |
| Google Analytics Input..... | 80 |
| Google Docs Input..... | 81 |
| Table Input..... | 82 |
| Text File Input..... | 83 |
| JMS Consumer..... | 88 |
| JMS Producer..... | 89 |
| Table Output..... | 90 |
| Text File Output..... | 91 |
| Select Values..... | 93 |
| Dummy (do nothing)..... | 94 |
| Filter Rows..... | 94 |
| Database Lookup..... | 95 |
| Stream Lookup..... | 96 |
| Web Services Lookup..... | 97 |
| Join Rows (Cartesian Product)..... | 97 |
| Merge Rows..... | 98 |
| Combination Lookup/Update..... | 99 |
| Dimension Lookup/Update..... | 101 |
| Group By..... | 104 |
| Modified JavaScript Value..... | 105 |
| Hadoop File Input..... | 106 |
| Hadoop File Output..... | 110 |
| Map/Reduce Input..... | 112 |

| | |
|---|------------|
| Map/Reduce Output..... | 113 |
| S3 File Output..... | 113 |
| RSS Input..... | 115 |
| HTTP Post..... | 117 |
| Job Step Reference..... | 119 |
| Start..... | 120 |
| Dummy..... | 120 |
| Job..... | 120 |
| Transformation..... | 121 |
| Mail..... | 122 |
| File Exists..... | 124 |
| Table Exists..... | 124 |
| JavaScript..... | 124 |
| Shell..... | 125 |
| SQL..... | 126 |
| HTTP..... | 126 |
| Get a File with FTP..... | 126 |
| Get a file with SFTP..... | 127 |
| Hadoop Copy Files..... | 128 |
| Hadoop Job Executor..... | 129 |
| Hadoop Transformation Job Executor..... | 131 |
| Amazon EMR Job Executor..... | 132 |
| Troubleshooting..... | 134 |
| Changing the Pentaho Data Integration Home Directory Location (.kettle folder)..... | 134 |
| Changing the Kettle Home Directory within the Pentaho BI Platform..... | 135 |
| FAQ: Pentaho Data Integration 4.1..... | 136 |

Introduction

Pentaho Data Integration is a flexible tool that allows you to collect data from disparate sources such as databases, files, and applications, and turn the data into a unified format that is accessible and relevant to end users. Pentaho Data Integration provides the Extraction, Transformation, and Loading (ETL) engine that facilitates the process of capturing the right data, cleansing the data, and storing the data using a uniform and consistent format.

Pentaho Data Integration provides support for slowly changing dimensions, (see Note below), and surrogate key for data warehousing, allows data migration between databases and application, is flexible enough to load giant datasets, and can take full advantage of cloud, clustered, and massively parallel processing environments. You can cleanse your data using transformation steps that range from very simple to very complex. Finally, you can leverage ETL as the data source for Pentaho Reporting.



Note: Dimension is a data warehousing term that refers to logical groupings of data such as product, customer, or geographical information. **Slowly Changing Dimensions (SCD)** are dimensions that contain data that changes slowly over time. For example, in most instances, employee job titles change slowly over time.

Common Uses of Pentaho Data Integration Include:

- Data migration between different databases and applications
- Loading huge data sets into databases taking full advantage of cloud, clustered and massively parallel processing environments
- Data Cleansing with steps ranging from very simple to very complex transformations
- Data Integration including the ability to leverage real-time ETL as a data source for Pentaho Reporting
- Data warehouse population with built-in support for slowly changing dimensions and surrogate key creation (as described above)

Audience and Assumptions

This guide is written for IT managers, database administrators, and Business Intelligence solution architects who have intermediate to advanced knowledge of ETL and Pentaho Data Integration Enterprise Edition features and functions.

You must have an installation of Pentaho Data Integration 4.1, (or later), if you want to examine some of the step-related information included in this document.

If you are new to Pentaho Data Integration

If you are novice user, Pentaho recommends that you start by following the exercises in *Getting Started with Pentaho Data Integration* available in the Pentaho Knowledge Base. You can return to this document when you have mastered some of the basic skills required to work with Pentaho Data Integration.

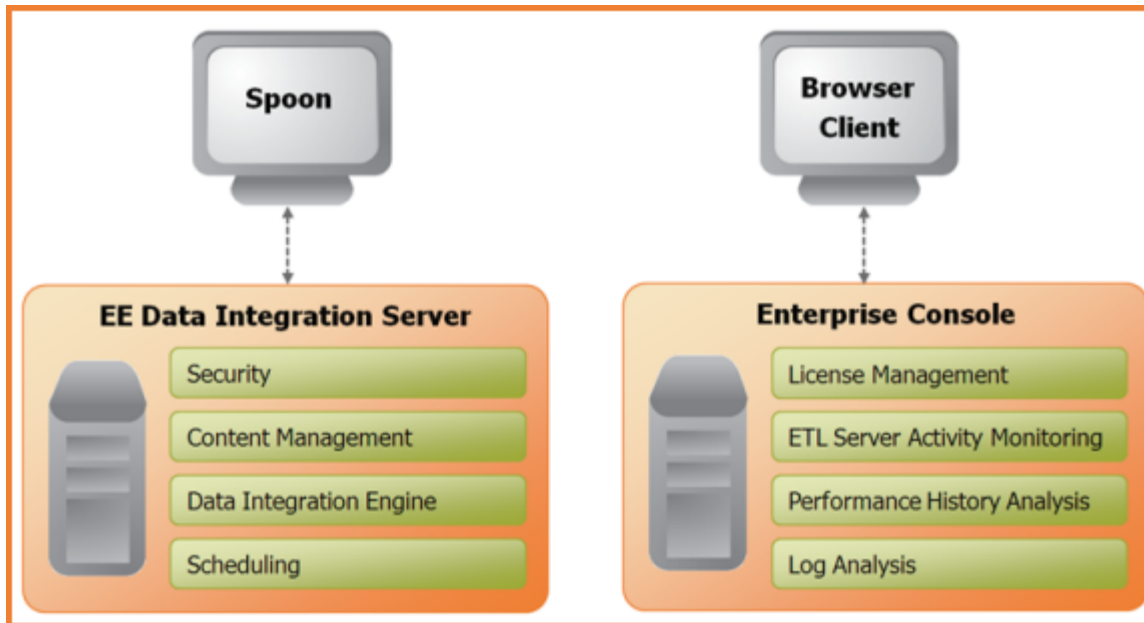
What this Guide Covers

This document provides you with information about the most *commonly used* steps. There are over 140 transformation steps included in Pentaho Data Integration. Future versions of this guide will include more coverage of PDI steps. Also included in this document is information about step performance, PDI architecture, using and managing variables, and more.

Refer to the *Pentaho Data Integration Administrator's Guide* and the *Pentaho Security Guide* for information about administering Pentaho Data Integration and configuring security related to LDAP and MSAD, respectively.

Pentaho Data Integration Architecture

The diagram below depicts the the core components of Pentaho Data Integration Enterprise Edition



Spoon is the design interface for building ETL jobs and transformations. Spoon provides a drag and drop interface allowing you to graphically describe what you want to take place in your transformations which can then be executed locally within Spoon, on a dedicated Data Integration Server, or a cluster of servers.

The **Data Integration Server** is a dedicated ETL server whose primary functions are:

| | |
|---------------------------|--|
| Execution | Executes ETL jobs and transformations using the Pentaho Data Integration engine |
| Security | Allows you to manage users and roles (default security) or integrate security to your existing security provider such as LDAP or Active Directory |
| Content Management | Provides a centralized repository that allows you to manage your ETL jobs and transformations. This includes full revision history on content and features such as sharing and locking for collaborative development environments. |
| Scheduling | Provides the services allowing you to schedule and monitor activities on the Data Integration Server from within the Spoon design environment. |

The **Enterprise Console** provides a thin client for managing deployments of Pentaho Data Integration Enterprise Edition including management of Enterprise Edition licenses, monitoring and controlling activity on a remote Pentaho Data Integration server and analyzing performance trends of registered jobs and transformations.

Pentaho Data Integration Components

Pentaho Data Integration is composed of the following primary components:

- **Spoon.** Introduced earlier, Spoon is a desktop application that uses a graphical interface and editor for transformations and jobs. Spoon provides a way for you to create complex ETL jobs without having to read or write code. When you think of Pentaho Data Integration as a product, Spoon is what comes to mind because, as a database developer, this is the application on which you will spend most of your time. Any time you author, edit, run or debug a transformation or job, you will be using Spoon.
- **Pan.** A standalone command line process that can be used to execute transformations and jobs you created in Spoon. The data transformation engine Pan reads data from and writes data to various data sources. Pan also allows you to manipulate data.

- **Kitchen.** A standalone command line process that can be used to execute jobs. The program that executes the jobs designed in the Spoon graphical interface, either in XML or in a database repository. Jobs are usually scheduled to run in batch mode at regular intervals.
- **Carte.** Carte is a lightweight Web container that allows you to set up a dedicated, remote ETL server. This provides similar remote execution capabilities as the Data Integration Server, but does not provide scheduling, security integration, and a content management system.

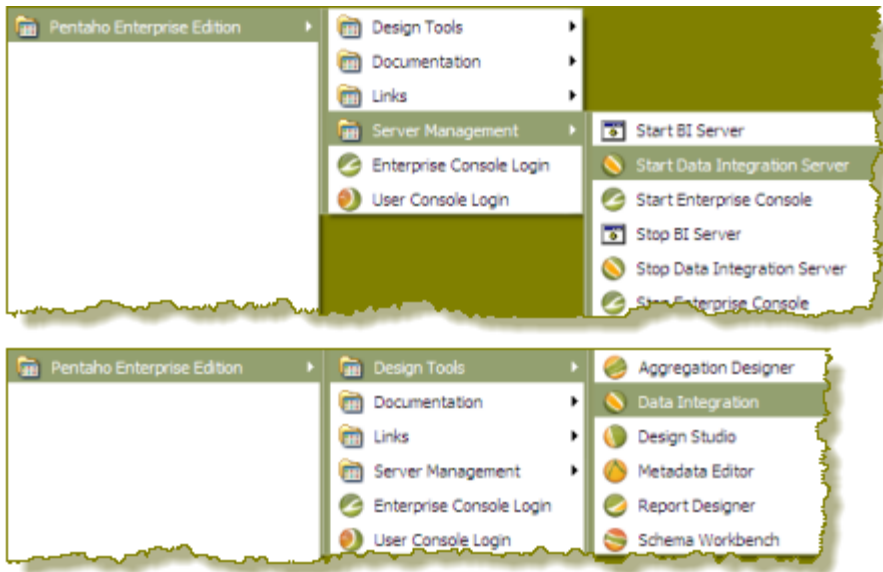
What's with all the Culinary Terms?

If you are new to Pentaho, you may sometimes see or hear Pentaho Data Integration referred to as, "Kettle." To avoid confusion, all you must know is that Pentaho Data Integration began as an open source project called, "Kettle." The term, K.E.T.T.L.E is a recursive that stands for **Kettle Extraction Transformation Transport Load Environment**. When Pentaho acquired Kettle, the name was changed to **Pentaho Data Integration**. Other PDI components such as Spoon, Pan, and Kitchen, have names that were originally meant to support a "restaurant" metaphor of ETL offerings.

Starting Pentaho Data Integration

The root directory of your Pentaho Data Integration installation, `\pdi-ee`, contains a set of scripts that make it convenient to start or stop all of the core server modules including the **Data Integration Server** and **Enterprise Console**.

If you are using Pentaho Data Integration as part of a BI Server installation in Windows, use **Start -> Programs** to launch the BI and Data Integration servers and the design tool (Data Integration/Spoon).




File paths will also be different depending on which method of installation was used. File paths associated with the installer, look like the following examples: ... \pentaho\server\data-integration-server, ... \pentaho\design-tools\data-integration.

File paths associated with a .zip or .tar installation look like the following example: ... \pdi-ee\data-integration.

Starting the Pentaho Data Integration Servers

To start Pentaho Data Integration servers...

1. Navigate to the folder where you have installed Pentaho Data Integration; for example, ... \Program Files \pdi-ee.
2. Double-click **start-servers.bat** to start the servers.

 **Note:** If you are using Linux or Macintosh, double-click **start-servers.sh**.

Starting Spoon

Follow the directions below to start Spoon, the Pentaho Data Integration graphical client tool.

1. Navigate to the folder where you have installed Pentaho Data Integration.

```
cd /pentaho/design-tools/data-integration/
```

2. Execute the **Spoon.bat** script (Windows) or **spoon.sh** (Linux, OS X) to start Spoon.

The Data Integration client tool will start. If you have not installed an Enterprise Edition license key, you will be prompted to do so.

Installing and Managing Enterprise Edition License Keys

If you do not have a valid PDI Enterprise Edition license key installed, Spoon will display a license management dialog box where you can view or remove existing Pentaho licenses, and install new ones. Expired licenses are displayed with a red mark to the left of the license title; valid licenses display a green checkmark.

To remove a license, select it, then click the red **X** in the upper right corner of the license list.


To add a license, click the round green **+** icon in the upper right corner of the window, then navigate to your Pentaho PDI Enterprise Edition and/or BI Suite for Hadoop license files. Only files with a .LIC extension will appear in the file dialogue.

You can also view, install, or remove licenses through the Pentaho Enterprise Console, or via the command line with the **install_license** script in the `/license-installer/` directory.

If you want to go to the PDI license manager in the future, start Spoon and go to the **Help** menu, then select **Register**.

Connecting to the Repository

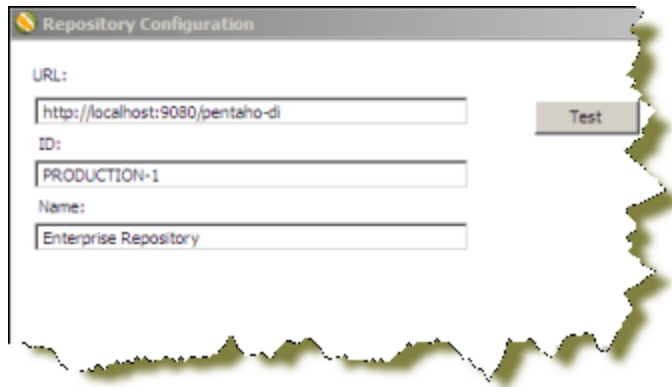
Each time you launch the designer (Spoon), the Repository Connection dialog box appears requesting you to log into the Pentaho Enterprise Repository. The Pentaho Enterprise Repository provides you a place to centrally store ETL jobs and transformations. Before you begin, you must have a user name and password to access the repository.

 **Note:** In a production environment, user access to Pentaho Data Integration is most likely established using LDAP or a custom authentication server. See the *Pentaho Security Guide* for information regarding LDAP and MSAD setup for Pentaho Data Integration.

If you do not want to see the Repository Connection dialog box every time you open the designer, disable the **Show this dialog at startup** check box.

To create a connection to the Enterprise Repository...

1. In the **Repository Connection** dialog box, click  (Add).
2. Select **Enterprise Repository:Enterprise Repository** and click **OK**. The **Repository Configuration** dialog box appears.
3. Enter the URL associated with your repository. Enter an ID and name for your repository.



4. Click **Test** to ensure your connection is properly configured. If you see an error message, make sure you started your [Data Integration Server](#) and that the **Repository URL** is correct.
5. Click **OK** to exit the **Success** dialog box.
6. Click **OK** to exit the Repository Configuration dialog box. Your new connection appears in the list of available repositories.
7. Enter your credentials, user name and password, for the Pentaho Enterprise Repository and click **OK**

Storing Content Alternatives

Pentaho Data Integration provides you with two ways of storing your transformations, jobs, and database connections as described below:

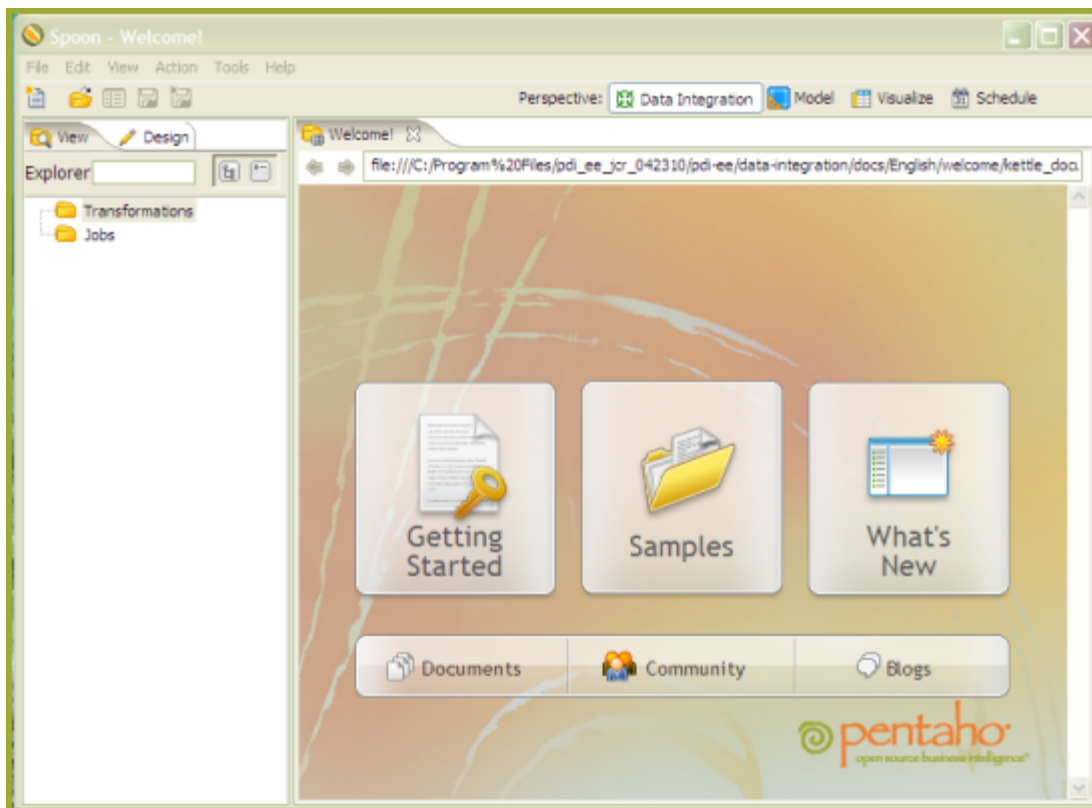
- **Pentaho Enterprise Repository** — You can save your jobs, transformations, and database connections in the Pentaho Enterprise Repository which provides you with content management, collaborative development, and enhanced security.
- **File-Based** — If you are not part of a collaborative team and do not want the overhead associated with Pentaho Enterprise Repository, you can save your jobs and transformations as files on your local device. Your database connection information is saved with your job or transformation. If you select this option, your jobs (.kjb) and transformations (.ktr) are saved in XML format.



Note: If you are an existing Pentaho Data Integration customer, see the *Pentaho Data Integration Upgrade Guide* for upgrade instructions.

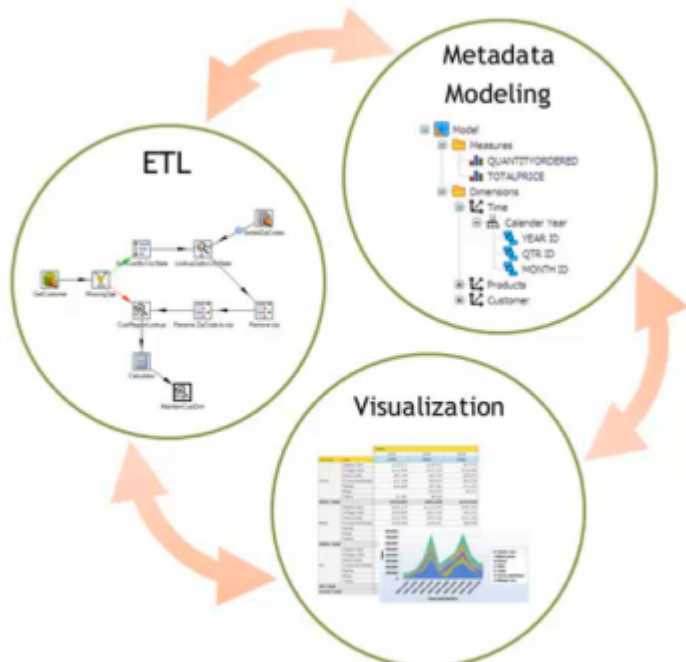
PDI Interface Perspectives

The **Welcome** page contains useful links to documentation, community links for getting involved in the Pentaho Data Integration project, and links to blogs from some of the top contributors to the Pentaho Data Integration project.



Introducing Perspectives

Pentaho Data Integration provides you with tools that include ETL, modeling, and visualization in one unified environment — the Spoon interface. This integrated environment allows you, as BI developer, to work in close cooperation with business users to build business intelligence solutions more quickly and efficiently.



When you are working in Spoon you can *change perspectives*, or switch from designing ETL jobs and transformations to modeling your data, and visualizing against the data. As users provide you with feedback about how the data is presented to them, you can quickly make iterative changes to your data directly in Spoon by changing perspectives. The ability to quickly respond to feedback and to collaborate with business users is part of the Pentaho Agile BI initiative. See the [Agile BI Techcast](#) series to learn more.



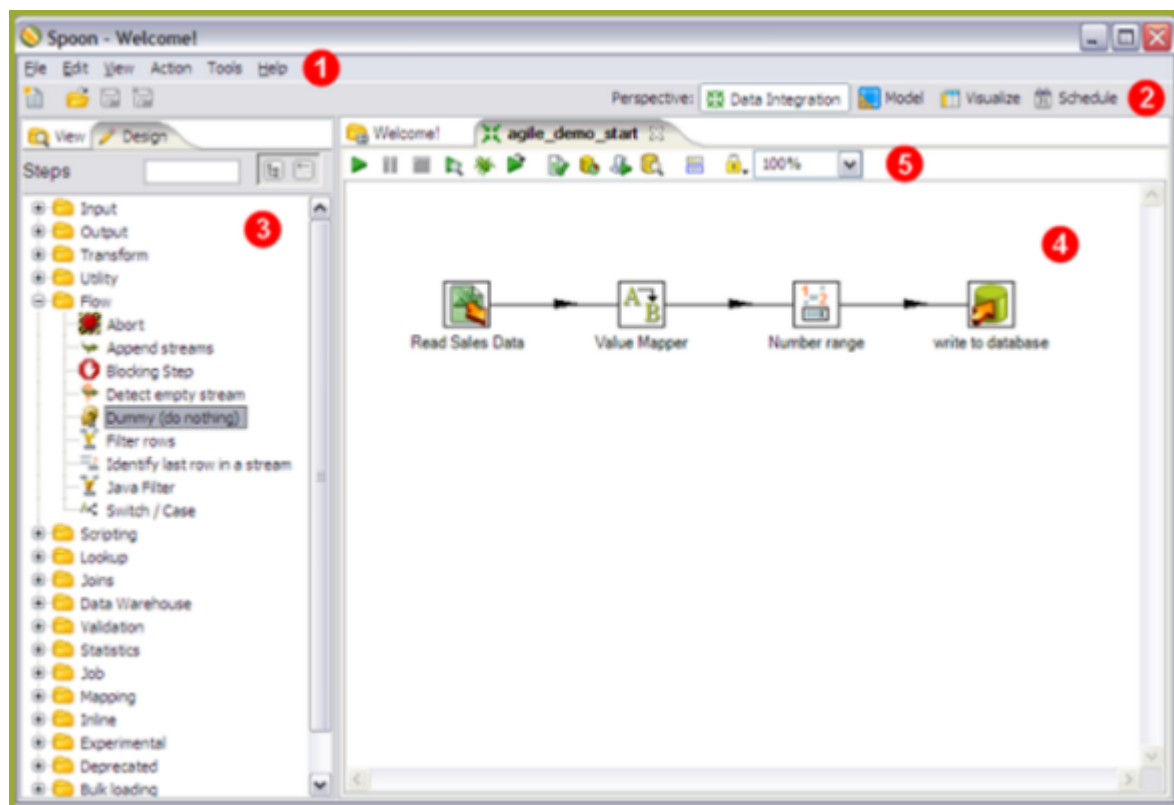
There are four perspectives in Spoon, they include:

- The Data Integration (ETL)
- Model
- Visualize
- Schedule (associated with the Data Integration perspective)

Each perspective is described in the sections that follow.

Data Integration (ETL) Perspective

The **Data Integration** perspective is used to design, preview, and test ETL jobs and transformations.




















The Data Integration perspective is organized into the components described in the table below:

| Component Name | Description |
|-----------------------|--|
| 1-Menubar | The Menubar provides access to common features such as properties, actions and tools |
| 2-Main Toolbar | <p>The Main Toolbar provides single-click access to common actions such as create a new file, opening existing documents, save and save as. The Data Integration perspective (shown in the image above), is used to create ETL transformations and jobs.</p> <p>The Schedule perspective (not shown) is related to the Data Integration perspective and is used to manage scheduled ETL activities on a Data Integration Server.</p> |

| Component Name | Description |
|------------------------------|--|
| 3-Design Palette | While in the Data Integration perspective, the Design Palette provides an organized list of transformation steps or job entries used to build transformations and jobs. Transformations are created by simply dragging transformation steps from the Design Palette onto the Graphical Workspace, or canvas, (4) and connecting them with hops to describe the flow of data. |
| 4-Graphical Workspace | The Graphical Workspace, or canvas, is the main design area for building transformations and jobs describing the ETL activities you want to perform. |
| 5-Sub-toolbar | The Sub-toolbar provides buttons for quick access to common actions specific to the transformation or job such as Run, Preview, and Debug. |

Toolbar Icons in the Data Integration Perspective


| Icon | Description |
|---|--|
|  | Create a new job or transformation |
|  | Open transformation/job from file if you are not connected to a repository or from the repository if you are connected to one |
|  | Explore the repository |
|  | Save the transformation/job to a file or to the repository |
|  | Save the transformation/job under a different name or file name (Save as) |
|  | Run transformation/job; runs the current transformation from XML file or repository |
|  | Pause transformation |
|  | Stop transformation |
|  | Preview transformation: runs the current transformation from memory. You can preview the rows that are produced by selected steps. |
|  | Run the transformation in debug mode; allows you to troubleshoot execution errors |
|  | Replay the processing of a transformation |
|  | Verify transformation |
|  | Run an impact analysis on the database |
|  | Generate the SQL that is needed to run the loaded transformation. |
|  | Launch the database explorer allowing you to preview data, run SQL queries, generate DDL and more |
|  | Hide execution results pane |
|  | Lock transformation |

VFS File Dialogues in Spoon

Some job and transformation steps have virtual filesystem (VFS) dialogues in place of the traditional local filesystem windows. VFS file dialogues enable you to specify a VFS URL in lieu of a typical local path. The following PDI and PDI plugin steps have such dialogues:

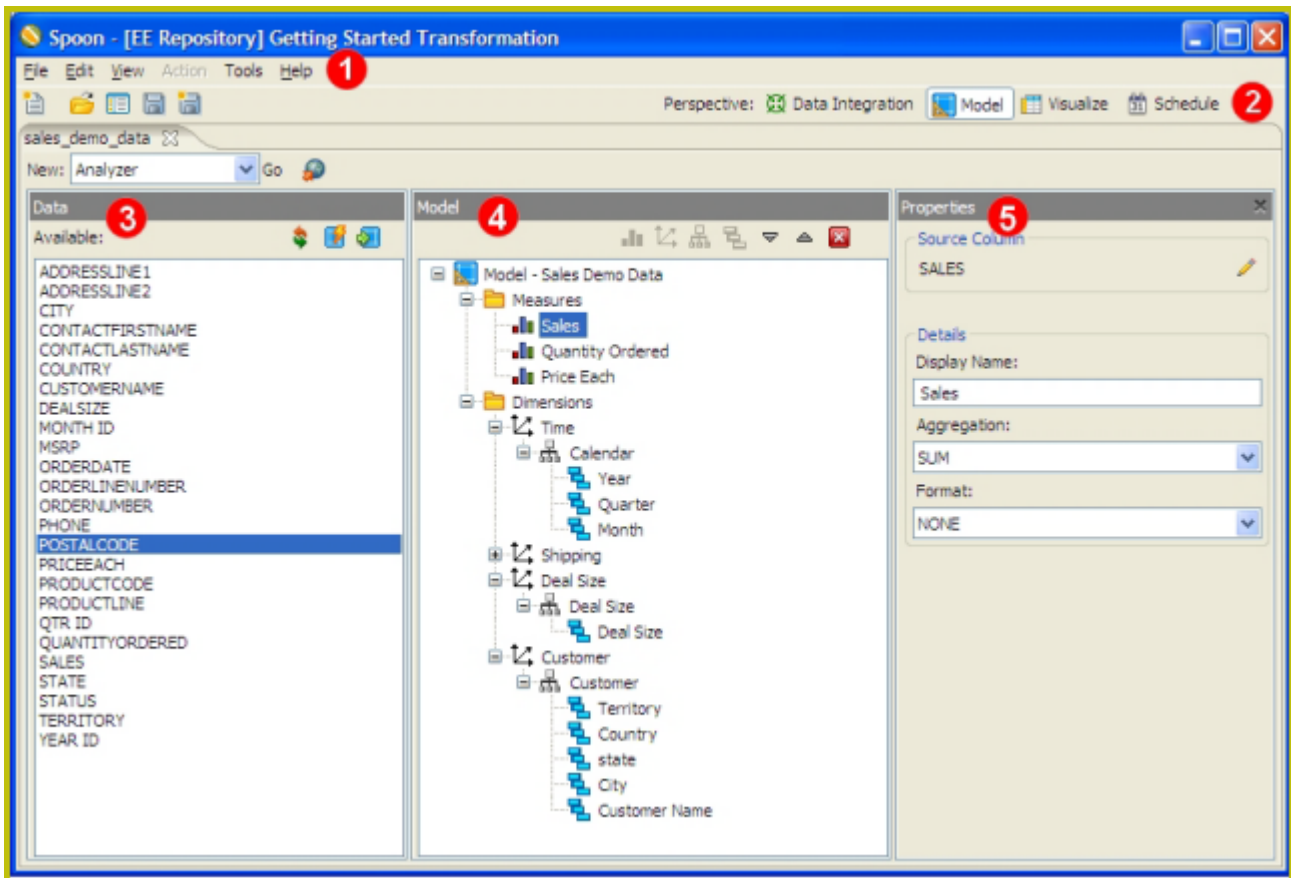
- File Exists
- Mapping (sub-transformation)

- ETL Meta Injection
- Hadoop Copy Files
- Hadoop File Input
- Hadoop File Output

 **Note:** VFS dialogues are configured through certain transformation parameters. Refer to [Configuring SFTP VFS](#) on page 41 for more information on configuring options for SFTP.

Modeling Perspective

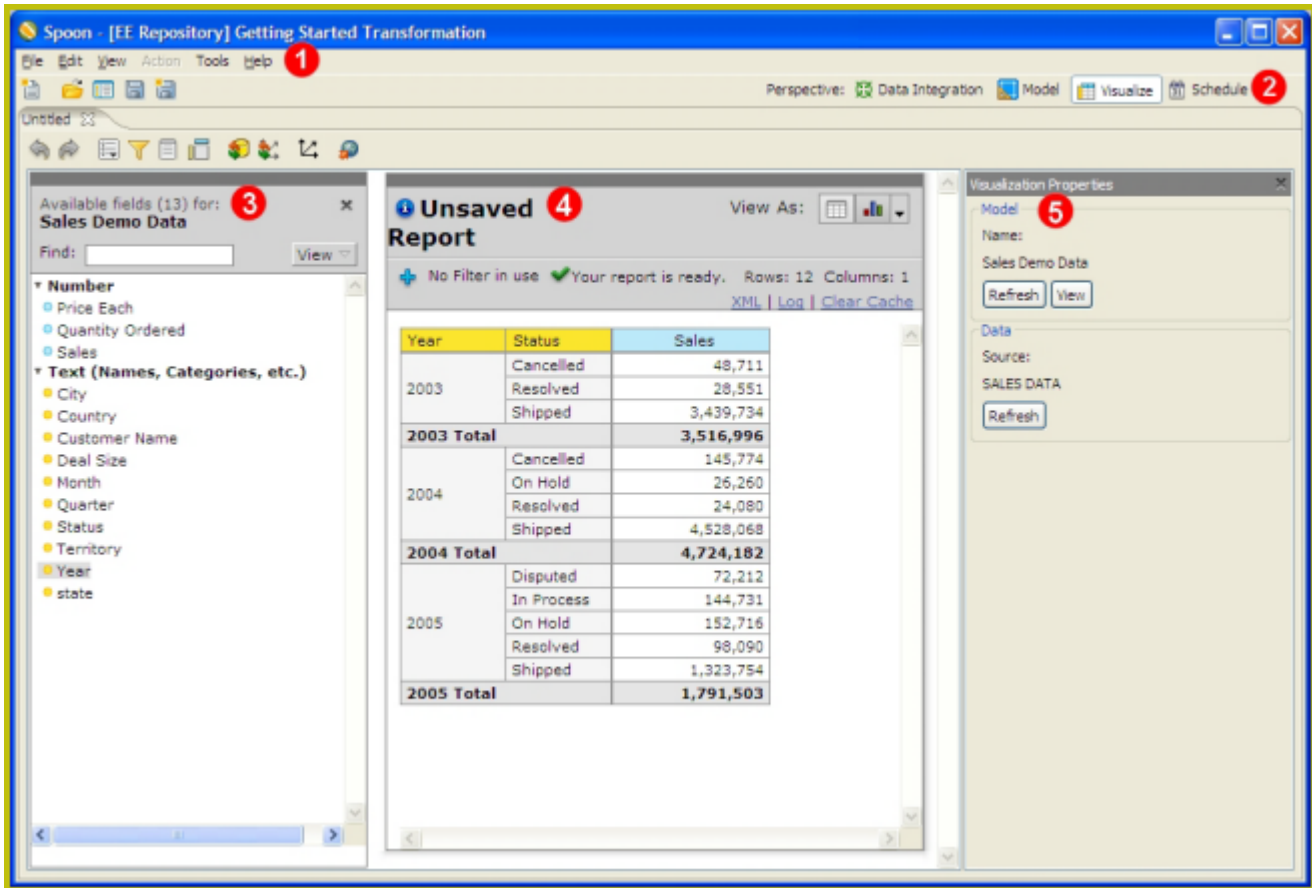
The **Model** perspective is used for designing reporting and OLAP metadata models that can be tested from within the **Visualize** perspective or published to the Pentaho BI Server.



| Component Name | Description |
|---------------------------|--|
| 1-Menuubar | The Menuubar provides access to common features such as properties, actions and tools |
| 2-Main Toolbar | The Main Toolbar provides single-click access to common actions such as create a new file, opening existing documents, save and save as. The right side of the main toolbar is also where you can switch between perspectives. |
| 3-Data Panel | Contains a list of available fields from your data source that can be used either as measure or dimension levels (attributes) within your OLAP dimensional model. |
| 4- Model Panel | Used to create measures and dimensions of your Analysis Cubes from the fields in the data panel. Create a new measure or dimension by dragging a field from the data panel over onto the Measures or Dimension folder in the Model tree. |
| 5-Properties Panel | Used to modify the properties associated with the selection in the Model Panel tree. |

Visualization Perspective

The **Visualize** perspective allows you to test reporting and OLAP metadata models created in the **Model** perspective using the Report Design Wizard and Analyzer clients respectively.



| Component Name | Description |
|-----------------------------------|--|
| 1-Menu Bar | The Menu Bar provides access to common features such as properties, actions and tools |
| 2-Main Toolbar | The Main Toolbar provides single-click access to common actions such as create a new file, opening existing documents, save and save as. The right side of the main toolbar is also where you can switch between perspectives. |
| 3-Field List | Contains the list of measures and attributes as defined in your model. These fields can be dragged into the Report Area (4) to build your query. |
| 4-Report Area | Drag fields from the field list into the Report Area to build your query. Right click on a measure or level to further customize your report with sub-totals, formatting, and more. |
| 5-Visualization Properties | Used to modify the properties associated with the selection in the Model Panel tree. |

Customizing the Spoon Interface

Kettle Options allow you to customize properties associated with the behavior and look and feel of the Spoon interface. Examples include startup options such as whether or not to display tips and the Welcome page, and user interface options such as fonts and colors. To access the options, in the menu bar, go to **Tools -> Options...**

The tables below contain descriptions for options under the **General** and **Look & Feel** tabs, respectively. You may want to keep the default options enabled initially. As you become more comfortable using Pentaho Data Integration, you can set the options to better suit your needs.

General

| Option | Description |
|---|---|
| Default number of lines in preview dialog | Sets the default number of lines that are displayed in the preview dialog box in Spoon |
| Maximum nr of lines in the logging windows | Specifies the maximum limit of rows to display in the logging window |
| Central log line store timeout in minutes | no def given |
| Max number of lines in the log history views | Specifies the maximum limit of line to display in the log history views |
| Show tips at startup? | Sets the display of tips at startup |
| Show welcome page at startup? | Controls whether or not to display the Welcome page when launching Spoon |
| Use database cache? | Spoon caches information that is stored on the source and target databases. In some instances, caching causes incorrect results when you are making database changes. To prevent errors you can disable the cache altogether instead of clearing the cache every time. |
| Open last file at startup? | Loads the last transformation you used (opened or saved) from XML or repository automatically |
| Auto save changed files? | Automatically saves a changed transformation before running |
| Only show the active file in the main tree? | Reduces the number of transformation and job items in the main tree on the left by only showing the currently active file |
| Only save used connections to XML? | Limits the XML export of a transformation to the used connections in that transformation. This is helpful while exchanging sample transformations to avoid having all defined connections to be included. |
| Ask about replacing existing connections on open/import? | Requests permission before replacing existing database connections during import |
| Replace existing connections on open/import? | This is the action that takes place when there is no dialog box shown, (see previous option) |
| Show Save dialog? | Allows you to turn off the confirmation dialogs you receive when a transformation has been changed |
| Automatically split hops? | Disables the confirmation messages that launch when you want to split a hop |
| Show copy or distribute dialog? | Disables the warning message that appears when you link a step to multiple outputs. This warning message describes the two options for handling multiple outputs: 1. Distribute rows - destination steps receive the rows in turns (round robin) 2. Copy rows - all rows are sent to all destinations |
| Show repository dialog at startup? | Controls whether or not the Repository dialog box appears at startup |
| Ask user when exiting? | Controls whether or not to display the confirmation dialog when a user chooses to exit the application |
| Clear custom parameters (steps/plugin-ins) | Clears all parameters and flags that were set in the plug-in or step dialog boxes. |
| Display tool tips? | Controls whether or not to display tool tips for the buttons on the main tool bar. |

Look & Feel

| Option | Description |
|-----------------------------------|--|
| Fixed width font | The font that is used in the dialog boxes, trees, input fields, and more; click Edit to edit the font or Delete to return the font to its default value |
| Font on workspace | The font that is used in the Spoon interface; click Edit to edit the font or Delete to return the font to its default value |
| Font for notes | The font to use in notes that are displayed in Spoon; click Edit to edit the font or Delete to return the font to its default value |
| Background color | Sets the background color in Spoon and affects all dialog boxes, too; click Edit to edit the color or Delete to return the background color to its default value |
| Workspace background color | Sets the background color in the graphical view of Spoon; click Edit to edit the background color or Delete to return the background color to its default value |
| Tab color | The color that is being used to indicate tabs that are active/selected; click Edit to edit the tab color or Delete to return the color to its default value |
| Icon size in workspace | 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. |
| Line width on workspace | Affects the line width of the hops in the Spoon graphical view and the border around the step. |
| Shadow size on workspace | If this size is larger than 0, a shadow of the steps, hops, and notes is drawn on the canvas, making it look like the transformation floats above the canvas. |
| Dialog middle percentage | By default, a parameter is drawn at 35% of the width of the dialog box, counted from the left. You can change using this option in instances where you use unusually large fonts. |
| Canvas anti-aliasing? | Some platforms like Windows, OSX and Linux support anti-aliasing through GDI, Carbon or Cairo. Enable this option for smoother lines and icons in your graph view. If you enable the option and your environment does not work, change the value for option "EnableAntiAliasing" to "N" in file \$HOME/.kettle/.spoonrc (C:\Documents and Settings\ <user>\.kettle\.spoonrc on Windows)</user> |
| Use look of OS? | Enabling this option on Windows allows you to use the default system settings for fonts and colors in Spoon. On other platforms, the default is always enabled. |
| Show branding graphics | Enabling this option will draw Pentaho Data Integration branding graphics on the canvas and in the left hand side "expand bar." |
| Preferred Language | Specifies the preferred language setting. |
| Alternative Language | Specifies the alternative language setting. Because the original language in which Pentaho Data Integration was written is English, it is best to set this locale to English. |

Terminology and Basic Concepts

Before you can start designing transformations and jobs, you must have a basic understanding of the terminology associated with Pentaho Data Integration.

Transformations, Steps, and Hops


A **transformation** is a network of logical tasks called *steps*. Transformations are essentially *data flows*. In the example below, the database developer has created a transformation that reads a flat file, filters it, sorts it, and loads it to a relational database table. Suppose the database developer detects an error condition and instead of sending the data to a Dummy step, (which does nothing), the data is logged back to a table. The transformation is, in essence, a directed graph of a logical set of data transformation configurations. Transformation file names have a .ktr extension.




The two main components associated with transformations are **steps** and **hops**:

Steps are the building blocks of a transformation, for example a text file input or a table output. There are over 140 steps available in Pentaho Data Integration and they are grouped according to function; for example, input, output, scripting, and so on. Each step in a transformation is designed to perform a specific task, such as reading data from a flat file, filtering rows, and logging to a database as shown in the example above. Steps can be configured to perform the tasks you require.

Hops are data pathways that connect steps together and allow schema metadata to pass from one step to another. In the image above, it seems like there is a sequential execution occurring; however, that is not true. Hops determine the flow of data *through* the steps not necessarily the sequence in which they run. When you run a transformation, each step starts up in its own thread and pushes and passes data.

 **Note:** All steps are started and run in parallel so the initialization sequence is not predictable. That is why you cannot, for example, set a variable in a first step and attempt to use that variable in a subsequent step.

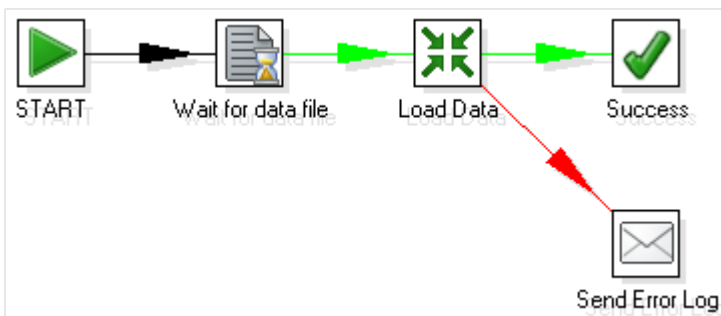
In Pentaho Data Integration 4.1, a new way to connect steps together, edit steps, and open the step contextual menu was added. Click  to edit your step. Click the down arrow to open the contextual menu. For information about connecting steps with hop, see [More About Hops](#).



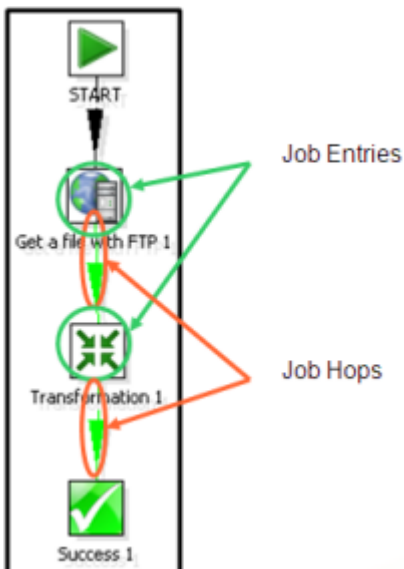
A step can have many connections — some join two steps together, some only serve as an input or output for a step. The data stream flows through steps to the various steps in a transformation. Hops are represented in Spoon as arrows. Hops allow data to be passed from step to step, and also determine the direction and flow of data through the steps. If a step sends outputs to more than one step, the data can either be copied to each step or distributed among them.

Jobs

Jobs are workflow-like models for coordinating resources, execution, and dependencies of ETL activities.



Jobs aggregate up individual pieces of functionality to implement an entire process. Examples of common tasks performed in a job include getting FTP files, checking conditions such as existence of a necessary target database table, running a transformation that populates that table, and e-mailing an error log if a transformation fails. The final job outcome might be a nightly warehouse update, for example.

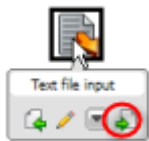


Jobs are composed of **job hops**, **job entries**, and **job settings**. Hops behave differently when used in a job, see [More About Hops](#). Job entries are the individual configured pieces as shown in the example above; they are the primary

building blocks of a job. In data transformations these individual pieces are called steps. Job entries can provide you with a wide range of functionality ranging from executing transformations to getting files from a Web server. A single job entry can be placed multiple times on the canvas; for example, you can take a single job entry such as a transformation run and place it on the canvas multiple times using different configurations. Job settings are the options that control the behavior of a job and the method of logging a job's actions. Job file names have a .kjb extension.

More About Hops

A hop connects one transformation step or job entry with another. The direction of the data flow is indicated by an arrow. To create the hop, click the source step, then press the <SHIFT> key down and draw a line to the target step. Alternatively, you can draw hops by hovering over a step until the hover menu appears. Drag the hop painter icon from the source step to your target step.



Additional methods for creating hops include:

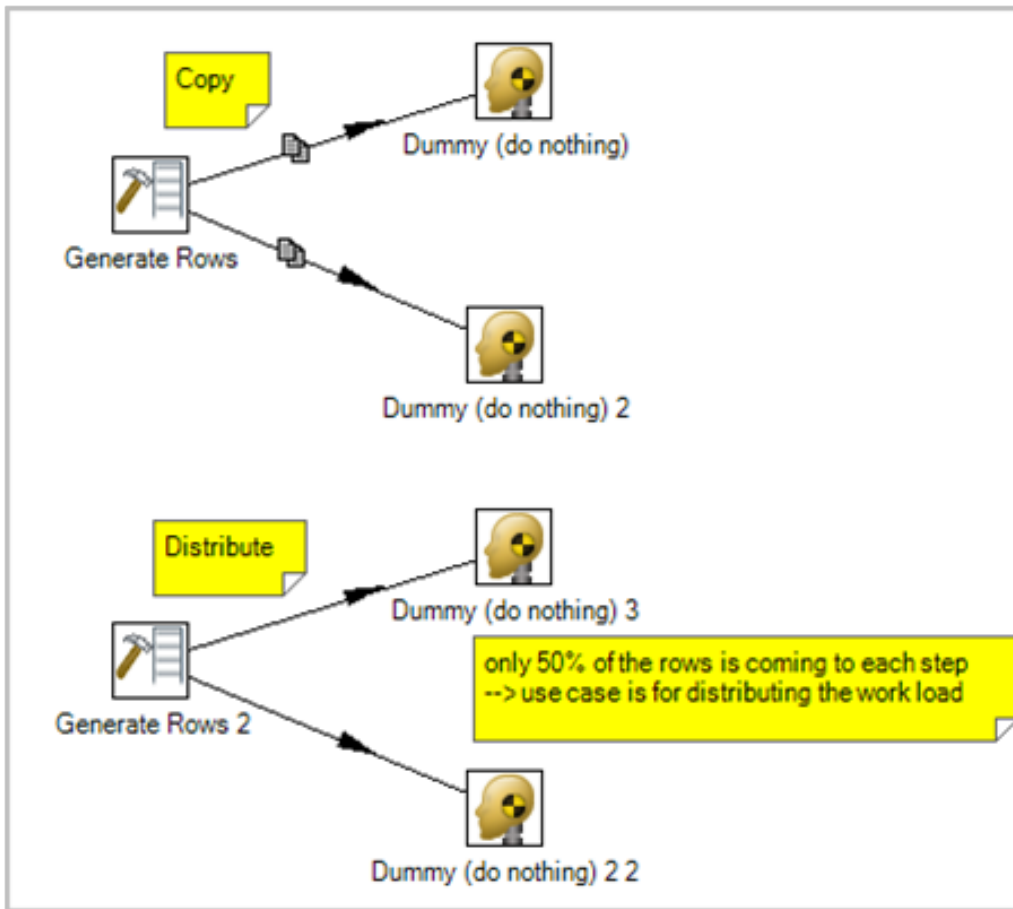
- Click on the source step, hold down the middle mouse button, and drag the hop to the target step.
- Select two steps, then choose New Hop from the right-click menu.
- Use <CTRL + left-click> to select two steps the right-click on the step and choose **New Hop**.

To **split a hop**, insert a new step into the hop between two steps by dragging the step over a hop. Confirm that you want to split the hop. This feature works with steps that have not yet been connected to another step only.

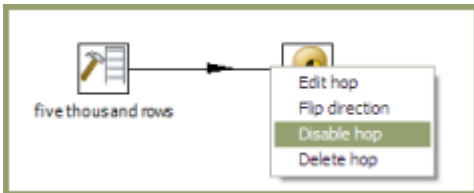
Loops are not allowed in transformations because Spoon depends heavily on the previous steps to determine the field values that are passed from one step to another. Allowing loops in transformations may result in endless loops and other problems. Loops are allowed in jobs because Spoon executes job entries sequentially; however, make sure you do not create endless loops.

Mixing rows that have a different layout is not allowed in a transformation; for example, if you have two table input steps that use a varying number of fields. Mixing row layouts causes steps to fail because fields cannot be found where expected or the data type changes unexpectedly. The trap detector displays warnings at design time if a step is receiving mixed layouts.

You can specify if data can either be **copied** or **distributed** between multiple hops leaving a step. Select the step, right-click and choose **Data Movement**.



A hop can be enabled or disabled (for testing purposes for example). Right-click on the hop to display the options menu.



Hop Colors in Transformations

Hops in transformations display in different colors based on the properties and state of the hop. The following table describes the meaning behind hop colors:

| Color | Meaning |
|---------------------|---|
| Green | Distribute rows: if multiple hops are leaving a step, rows of data will be evenly distributed to all target steps |
| Red | Copies rows: if multiple hops are leaving a step, all rows of data will be copied to all target steps |
| Yellow | Provides info for step, distributes rows |
| Gray | The hop is disabled |
| Black | The hop has a named target step |
| Blue | Candidate hop using middle button + drag |
| Red (Bold Dot line) | The hop is used for carrying rows that caused errors in source step(s). |

Job Hops

Besides the execution order, a hop 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. A job hop is just a flow of control. Hops link to job entries and, based on the results of the previous job entry, determine what happens next.



| Option | Description |
|------------------------------------|---|
| 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 be executed only when the result of the originating job entry is true; this means a successful execution such as, file found, table found, without error, and so on |
| 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, and so on |

Hop Colors in Jobs

Hops in jobs display in different colors based on the properties and state of the hop. The following table describes the meaning behind hop colors:


| Color | Meaning |
|--------------|---|
| Black | The target entry executes regardless of the result of the source entry (Unconditional) |
| Green | The target entry executes only if the result of the source entry is successful (Result is true) |
| Red | The target entry executes only if the source entry failed (Result is false) |

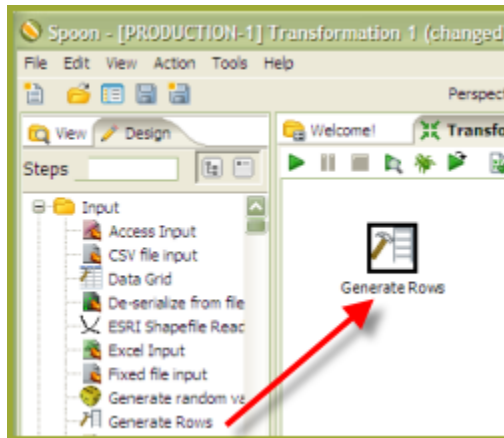
Creating your First Transformation


This exercise is designed to help you learn basic skills associated with handling steps and hops, running and previewing transformations. See the *Getting Started with Pentaho Data Integration* guide for a comprehensive, "real world" exercise for creating, running, and scheduling transformations.

Getting Started

Follow the instructions below to begin creating your transformation.

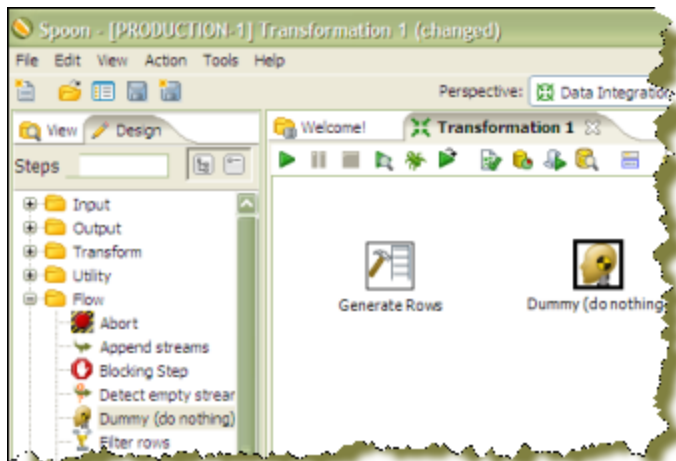
1. Click  (New) in the upper left corner of Spoon.
2. Select **Transformation** from the list.
3. Under the **Design** tab, expand the **Input** node; then, select and drag a **Generate Rows** step onto the canvas on the right.



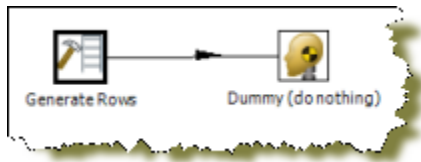
 **Note: (Tip)** If you don't know where to find a step, there is a search function in the left corner of Spoon. Type the name of the step in the search box. Possible matches appear under their associated nodes. Clear your search criteria when you are done searching.




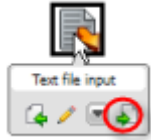
4. Expand the **Flow** node; click and drag a **Dummy (do nothing)** step onto the canvas.



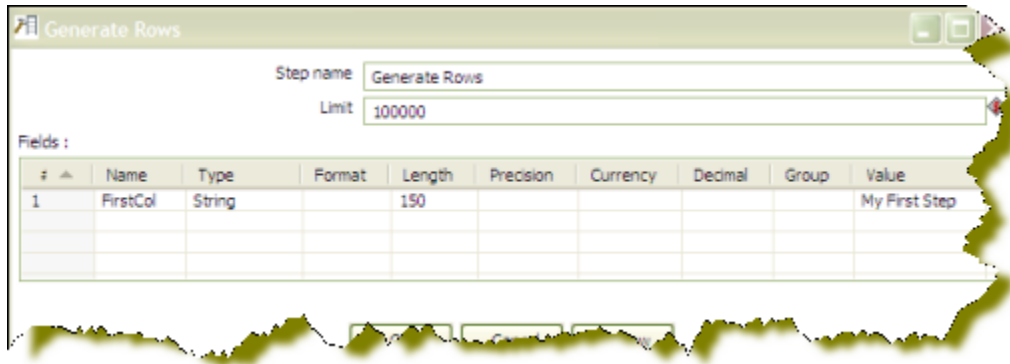
5. To connect the steps to each other, you must add a hop. Hops are used to describe the flow of data between steps in your transformation. To create the hop, click the **Generate Rows** step, then press and hold the **<SHIFT>** key then draw a line to the **Dummy (do nothing)** step.



 **Note:** Alternatively, you can draw hops by hovering over a step until the hover menu appears. Drag the hop painter icon from the source step to your target step.




6. Double click the **Generate Rows** step to open its edit properties dialog box.
7. In the **Limit** field, type 100000.
This limits the number of generated rows to 100,000.
8. Under **Fields:**, type **FirstCol** in the **Name** field.
9. Under **Type**, enter **String**.
10. Under **Value**, type **My First Step**. Your entries should look like the image below. Click **OK** to exit the Generate Rows edit properties dialog box.

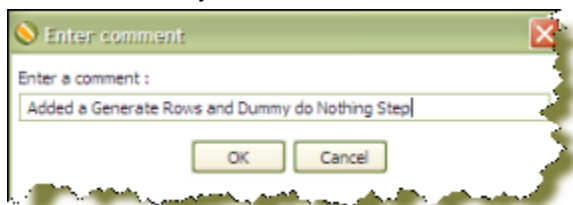


11. Now, save your transformation. See [Saving Your Transformation](#).

Saving Your Transformation

Follow the instructions below to save your transformation.

1. In Spoon, click **File** -> **Save As**.
The **Transformation Properties** dialog box appears.
2. In the **Transformation Name** field, type **First Transformation**.
3. In the **Directory** field, click  (folder icon) to select a repository folder where you will save your transformation.
4. Expand the **Home** directory and double-click the **joe** folder.
Your transformation will be saved in the **joe** folder in the Enterprise Repository.
5. Click **OK** to exit the **Transformation Properties** dialog box.
The **Enter Comment** dialog box appears.
6. Click in the **Enter Comment** dialog box and press <Delete> to remove the default text string. Type a meaningful comment about your transformation.



The comment and your transformation are tracked for version control purposes in the Enterprise Repository.

7. Click **OK** to exit the **Enter Comment** dialog box and save your transformation.

Running Your Transformation Locally

In the exercise, you created a simple transformation. Now, you are going to run your transformation locally (Local Execution). Local execution allows you to execute a transformation or job from within the Spoon design environment (on your local device). This is ideal for designing and testing transformations or lightweight ETL activities.

1. In Spoon, go to **File -> Open**.

The contents of the repository appear.


2. Navigate to the folder that contains your transformation.

If you are a user with administrative rights, you may see the folders of other users.

3. Double-click on your transformation to open it in the Spoon workspace.



Note: If you followed the exercise instructions, the name of the transformation is **First Transformation**.

4. In the upper left corner of the workspace, click  (Run).

The **Execute a Transformation** dialog box appears. Notice that **Local Execution** is enabled by default.

5. Click **Launch**.

The **Execution Results** appear in the lower pane.


6. Examine the contents under **Step Metrics**. The Step Metrics tab provides statistics for each step in your transformation such as how many records were read, written, caused an error, processing speed (rows per second) and more. If any of the steps caused the transformation to fail, they would be highlighted in red.

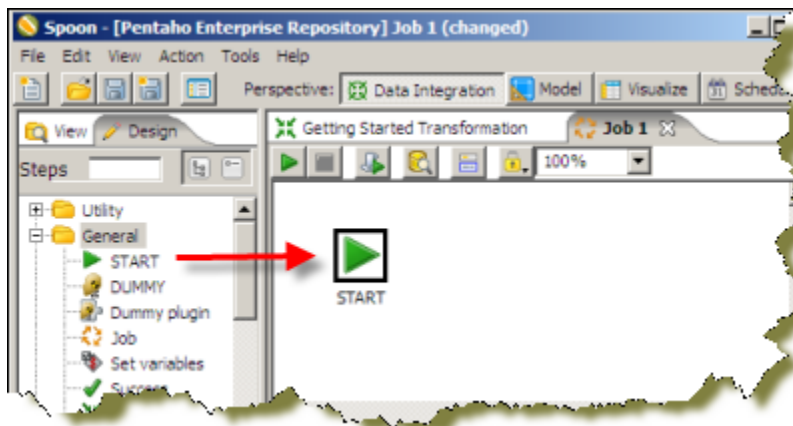


Note: Other tabs associated with Execution Results, require additional set up. See [Performance Monitoring and Logging](#).


Building a Job

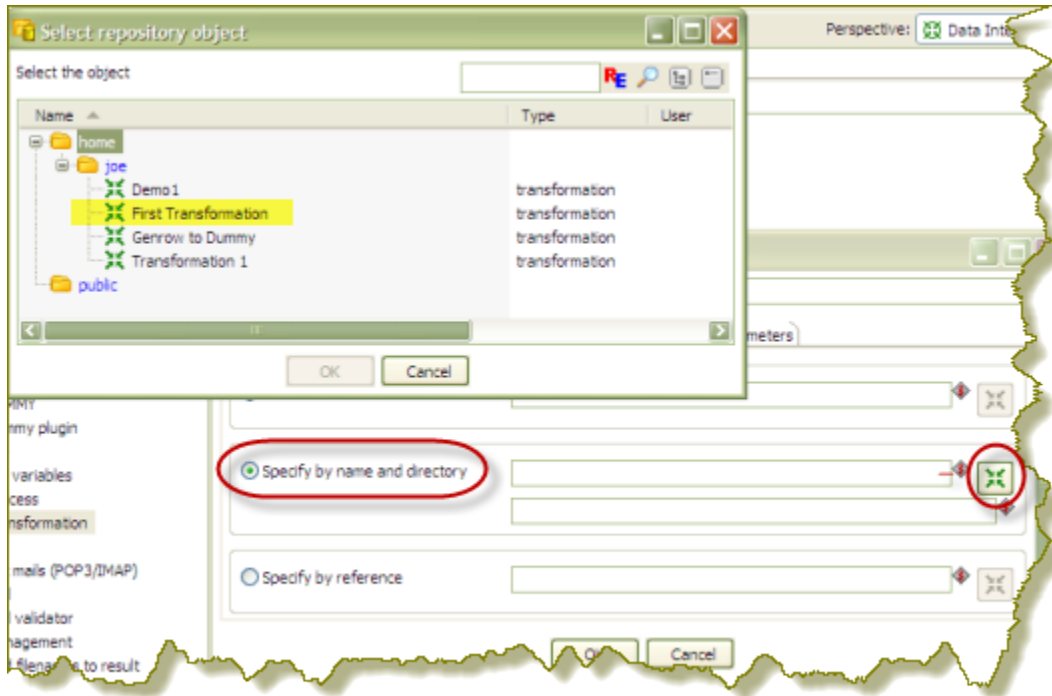
You created, saved, and ran your first transformation. Now, you will build a simple job. Use jobs to execute one or more transformations, retrieve files from a Web server, place files in a target directory, and more. Additionally, you can schedule jobs to run on specified dates and times. The *Getting Started with Pentaho Data Integration Guide* contains a "real world" exercise for building jobs.

1. In the Spoon menubar, go to **File -> New > Job**. Alternatively click  (New) in the toolbar.
2. Click the **Design** tab.
The nodes that contain job entries appear.
3. Expand the **General** node and select the **Start** job entry.
4. Drag the Start job entry to the workspace (canvas) on the right.




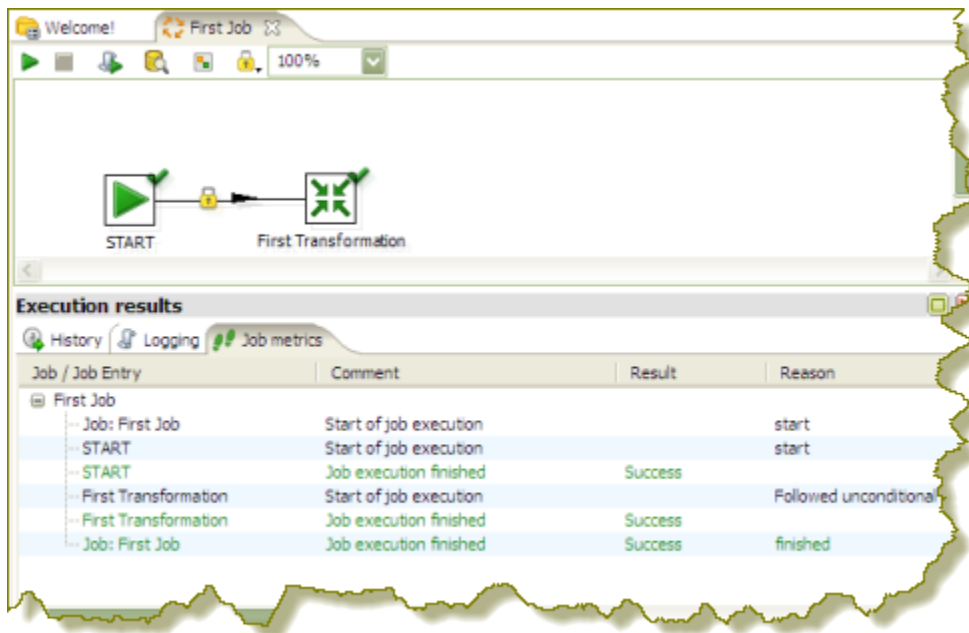
The Start job entry defines where the execution will begin.

5. Expand the **General** node, select and drag a **Transformation** job entry on to the workspace.
6. Use a hop to connect the Start job entry to the Transformation job entry.
7. Double-click on the **Transformation** job entry to open its properties dialog box.
8. Under **Transformation specification**, click **Specify by name and directory**.
9. Click  (Browse) to locate your transformation in the Enterprise Repository.
10. In the **Select repository object** view, expand the **Home** and **job** directories. Locate **First Transformation** and click **OK**.




The name of the transformation and its location appear next to the **Specify by name and directory** option.

11. Under **Transformation specification**, click **OK**.
12. Save your job; call it **First Job**. Steps used to save a job are nearly identical to saving a transformation. Provide a meaningful comment when saving your job. See [Saving Your Transformation](#).
13. Click  (Run Job) in the toolbar. When the **Execute a Job** dialog box appears, choose **Local Execution** and click **Launch**.

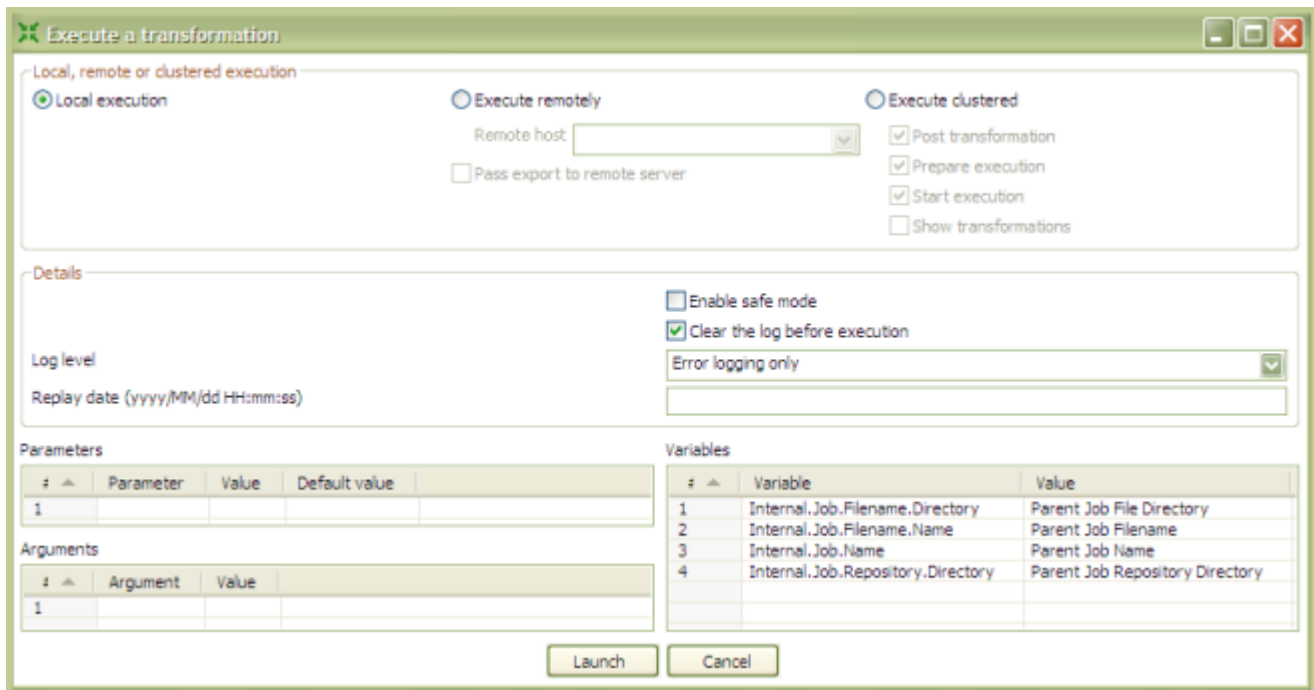


The **Execution Results** panel opens displaying the job metrics and log information for the job execution.

Executing Transformations

When you are done modifying a transformation or job, you can run it by clicking  (Run) from the main menu toolbar, or by pressing F9. There are three options that allow you to decide where you want your transformation to be executed:

- **Local Execution** — The transformation or job will be executed 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 the Data Integration Server running or Pentaho Data Integration installed on a remote machine and running the Carte service. To use remote execution you first must set up a slave server (see [Setting Up a Slave Server](#)).
- **Execute clustered** — Allows you to execute the job or transformation in a clustered environment.



| # | Parameter | Value | Default value |
|---|-----------|-------|---------------|
| 1 | | | |


| # | Variable | Value |
|---|-----------------------------------|---------------------------------|
| 1 | Internal.Job.Filename.Directory | Parent Job File Directory |
| 2 | Internal.Job.Filename.Name | Parent Job Filename |
| 3 | Internal.Job.Name | Parent Job Name |
| 4 | Internal.Job.Repository.Directory | Parent Job Repository Directory |

Setting Up a Slave Server

In order to proceed with these instructions, you must first install the PDI client tools on each slave machine. Refer to the *Pentaho Data Integration Administrator's Guide* for instructions for that process.

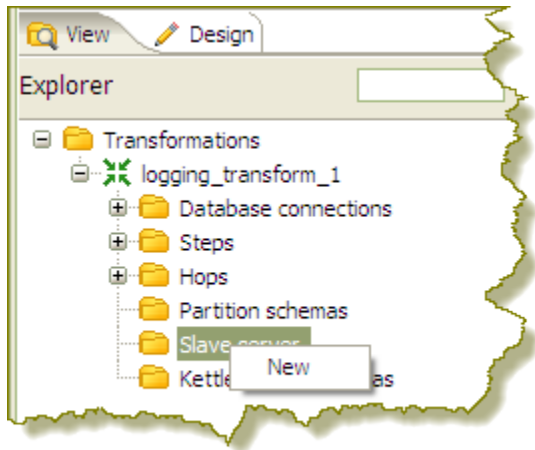
Large transformations can quickly overload your network and cause out-of-memory issues and other related problems. Pentaho Data Integration is capable of forwarding transformations to one or more dedicated servers, which can dramatically decrease execution time on workstations. Dedicated servers run on demand and allow you greater control over the execution of jobs and transformations. That is why they are referred to as *slave servers*.

If you have used previous versions of Pentaho Data Integration, you are familiar with Carte, a small Web server that can be installed on a remote device. In Pentaho Data Integration 4.1, you can continue using Carte or you can use one (or more) Data Integration slave servers.

 **Note:** Carte server instances are better suited for clustered environments. See the *Pentaho Data Integration Administrator's Guide* for details.

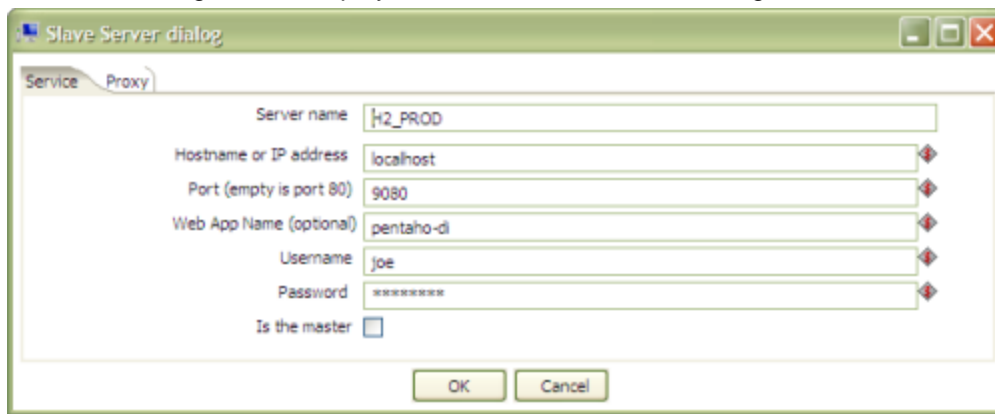
To set up a slave server...

1. Open a transformation.
2. In the **Explorer View** in Spoon, select **Slave Server**.
3. Right-click and select **New**.




The **Slave Server** dialog box appears.

- In the Slave Server dialog box, enter the appropriate connection information for the Data Integration (or Carte) slave server. The image below displays a connection to the Data Integration slave server.



| Option | Description |
|-------------------------------|--|
| Server name | The name of the slave server |
| Hostname or IP address | The address of the device to be used as a slave |
| Port | Defines the port you are for communicating with the remote server |
| Web App Name | Used for connecting to the DI server and set to pentaho-di by default |
| User name | Enter the user name for accessing the remote server |
| Password | Enter the password for accessing the remote server |
| Is the master | Enables this server as the master server in any clustered executions of the transformation |

 **Note:** When executing a transformation or job in a clustered environment, you should have one server set up as the master and all remaining servers in the cluster as slaves.

Below are the proxy tab options:

| Option | Description |
|---|---|
| Proxy server hostname | Sets the host name for the Proxy server you are using |
| The proxy server port | Sets the port number used for communicating with the proxy |
| Ignore proxy for hosts: regex 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. |

- Click **OK** to exit the dialog box. Notice that a plus sign (+) appears next to **Slave Server** in the Explorer View.

Executing Transformations and Jobs Remotely

Both transformations and jobs can be executed on remote servers. The options for remote execution of either a job or transformation is virtually identical.

Execute a transformation

Local, remote or clustered execution

Local execution Execute remotely Execute clustered

Remote host: H2_PROD

Pass export to remote server

Post transformation
 Prepare execution
 Start execution
 Show transformations

Details

Enable safe mode
 Clear the log before execution

Log level: Detailed logging

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

Parameters

| # | Parameter | Value | Default value |
|---|-----------|-------|---------------|
|---|-----------|-------|---------------|

Arguments

| # | Argument | Value |
|---|----------|-------|
|---|----------|-------|

Variables

| # | Variable | Value |
|---|---------------------------------|------------|
| 1 | Internal.Job.FileName.Directory | Parent Job |
| 2 | Internal.Job.FileName.Name | Parent Job |

Launch Cancel

Execute a job

Local or remote execution

Local execution Execute remotely

Remote host: H2_PROD

Pass export to remote server

Details

Enable safe mode
 Clear the log before execution

Log level: Detailed logging

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

Parameters

| # | Parameter | Value | Default value |
|---|-----------|-------|---------------|
| 1 | | | |

Arguments


| # | Argument | Value |
|---|----------|-------|
| 1 | 01 | |

Variables

| # | Variable | Value |
|---|----------|-------|
| 1 | | |

Launch Cancel

Follow the instructions below to perform a remote execution of a job or transformation.

1. Make sure the Data Integration server is running and that you have a transformation (or job) open.
2. Click  (Run) to open the Execute a Transformation (or job) dialog box.
3. Enable **Execute Remotely** and select the slave server (Data Integration or Carte) from the list of available servers (**Remote Host**).

Enable **Pass export to remote server** to ensure that a job or transformation can run on a remote server that does not have access to the required Pentaho Data Integration metadata. This option is available to facilitate cloud and grid computing. It works by exporting a job (or transformation) as well as all the underlying resources (transformations and jobs) to a single .zip archive. This .zip archive is then transferred prior to execution. The archive is never extracted as the job or transformation is executed directly from the archive.

Use **Enable Safe Mode** to have Pentaho Data Integration check to ensure that every row that is sent over a single hop has the same structure: field names, types, order of fields, and so on.

If appropriate, set the **Log Level** to specify the level of detail you want to capture in the log. (see [Performance Monitoring and Logging](#) for more information).

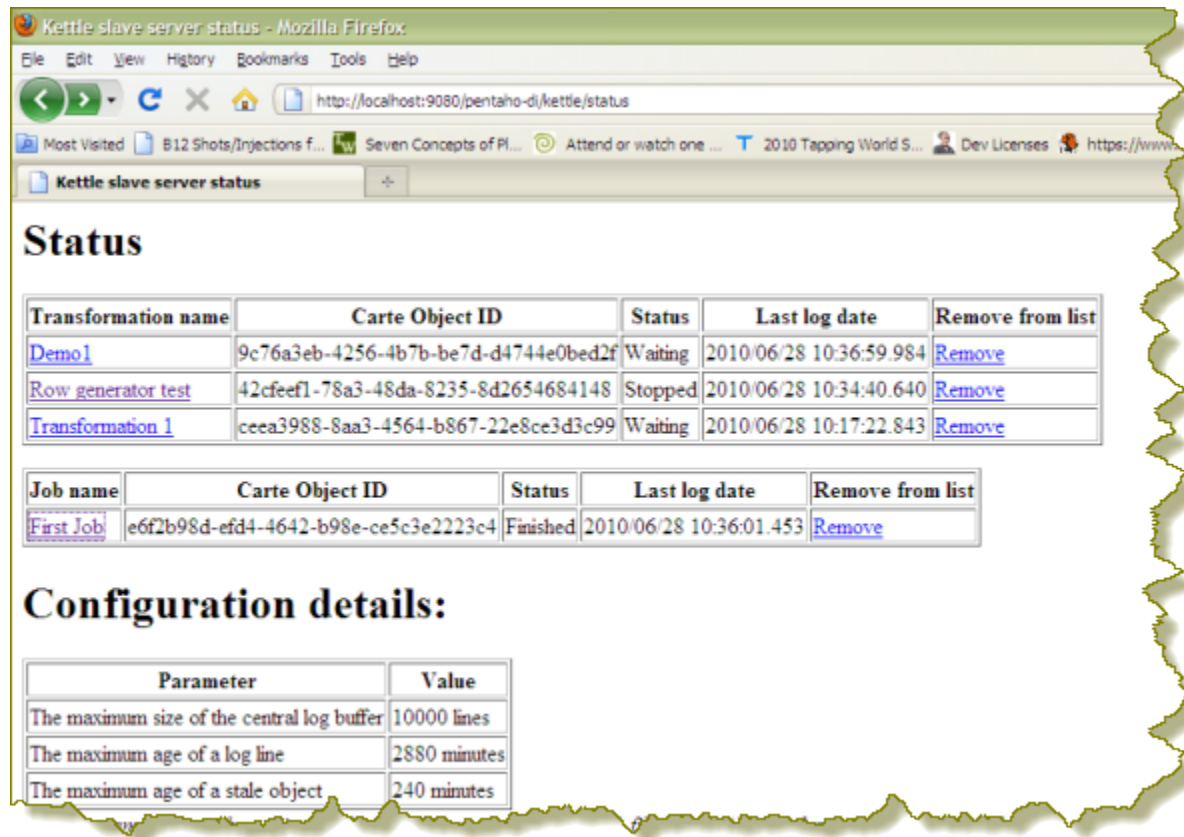
Replay date if you want to replay the transformation.

The **Parameters** grid allows you to set the value of parameters to be used when running the transformation (or job).

The **Arguments** grid allows you to set the value of arguments to be used when running the transformation (or job).

The **Variables** grid allows you to set the value of variables to be used when running the transformation (or job).

4. Click **Launch**.
5. Click the **Slave Server** tab to monitor your transformation.
6. Log onto the Data Integration server.
7. Click to select the transformation (or job) you want to monitor. The **Status** page appears allowing you to monitor the transformation (or job) as shown in the image below.



Creating a Cluster Schema

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.

Begin by selecting the **Kettle cluster schemas** node in the Spoon **Explorer View**. Right-click and select **New** to open the **Clustering Schema** dialog box.

| Option | Description |
|--------------------|---|
| Schema name | The name of the clustering schema |
| Port | 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: to avoid networking problems, make sure no other networking protocols are in the same range . |

| Option | Description |
|------------------------------------|--|
| Sockets buffer size | The internal buffer size to use |
| Sockets flush interval rows | The number of rows after which the internal buffer is sent completely over the network and emptied. |
| Sockets data compressed? | When enabled, all data is compressed using the Gzip compression algorithm to minimize network traffic |
| Dynamic cluster | The cluster schema where the slave servers are only known at runtime. Used in instances where hosts are being added or removed at will, such as in cloud computing settings |
| Slave Servers | A list of the servers to be used in the cluster. You must have one master server and any number of slave servers. To add servers to the cluster, click Select slave servers to select from the list of available slave servers. |

Executing Transformations in a Cluster

When running transformations in a clustered environment, you have the following options:

- **Post transformation** — Splits the transformation and post it to the different master and slave servers
- **Prepare execution** — Runs the initialization phase of the transformation on the master and slave servers
- **Prepare execution** — Runs the initialization phase of the transformation on the master and slave servers
- **Start execution** — Starts the actual execution of the master and slave transformations.
- **Show transformations** — Displays the generated (converted) transformations that will be executed on the cluster


See [Basic Clustering Example](#) and [Clustering and Clouds Made Easy](#) for additional information.

Impact Analysis

To see what effect your transformation will have on the data sources it includes, go to the **Action** menu and click on **Impact**. PDI will perform an impact analysis to determine how your data sources will be affected by the transformation if it is completed successfully.

Working in the Enterprise Repository


In addition to storing and managing your jobs and transformations, the Enterprise Repository provides full revision history for documents allowing you to track changes, compare revisions and revert to previous versions when necessary. This, in combination with other features such as enterprise security and content locking make the Enterprise Repository an ideal platform for providing a collaborative ETL environment.


 **Note:** If you prefer to manage your documents as loose files on the file system, click **Cancel** in the **Repository Connection** dialog box. You can also stop the Repository Connection dialog box from appearing at startup by disabling the **Show this dialog at startup** option.


Adding an Enterprise Repository

To add a new enterprise repository...

1. In the Spoon menu bar, go to **Tools -> Repository -> Connect**.

 **Note:** If you are currently connected to a repository you must click **Disconnect Repository** and return to this step to continue.


2. In the Repository Connection dialog box, click  (**Add**).
3. Select **Enterprise Repository** and click **OK**. The **Repository Configuration** dialog box appears.
4. In the **Repository Configuration** dialog box, enter the following values and click **OK**:

| Field Name | Description |
|-------------|---|
| URL | http://localhost:9080/pentaho-di — Click Test to make sure your repository URL is correct. In production, this URL may be different than the default URL provided here.  Note: To find out what port your repository (Data Integration server port) is on, locate the pentaho folder on your local device or network drive and open the installation-summary.text file. |
| ID | Enter a unique ID for this repository. |
| Name | Enter a name for your repository. |

5. Click **OK**. Notice that the new repository has been added to the list of available repositories in the **Repository Connection** dialog box.
6. Enter the appropriate credentials (user name and password) to access the repository and click **OK**.


Editing Enterprise Repository Details

To edit Enterprise Repository details...

1. In the **Repository Connection** dialog box, select the repository whose details you want to edit.
2. Click  (**Edit**).
The **Repository Configuration** dialog box appears.
3. Make changes as needed and click **OK** when you are done.

Deleting an Enterprise or Kettle Database Repository

When necessary, you can delete an Enterprise Repository or Kettle Database repository. To delete a repository...

1. In the **Repository Connection** dialog box, select the repository you want to delete from the list of available repositories.
2. Click  (**Delete**).


A confirmation dialog appears.

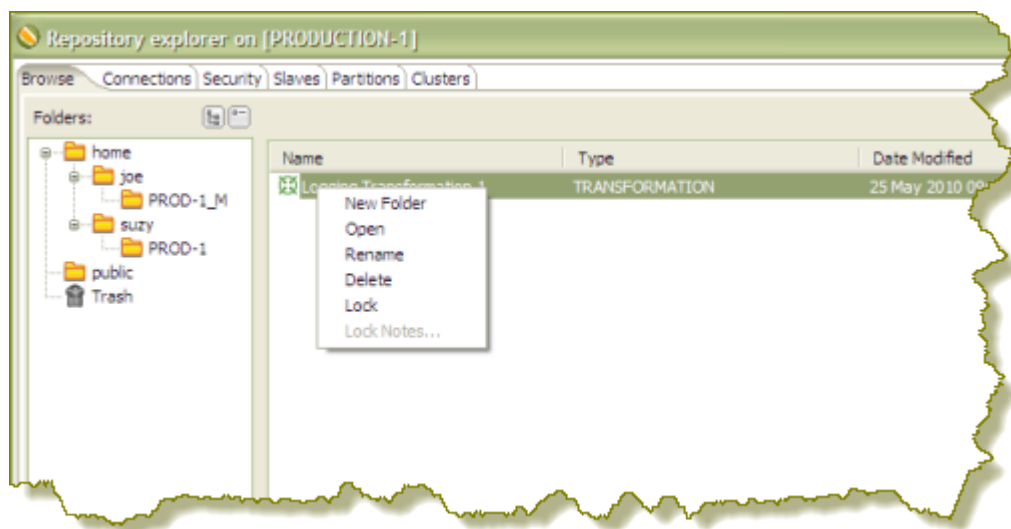
3. Click **Yes** to delete the repository.

Managing Content in the Enterprise Repository


When you are in the Repository Explorer view (**Tools -> Repository -> Explore**) use the right-click menu to perform common tasks such as those listed below:

- Exploring repository contents
- Sharing content with other repository users
- Creating a new folder in the repository
- Opening a folder, job, or transformation
- Renaming a folder, job or transformation
- Deleting a folder, job, or transformation
- Locking a job or transformation

 **Note:** Permissions set by your administrator determine what you are able to view and tasks you are able to perform in the repository.



To **move** objects, such as folders, jobs, or transformations, in the repository, select the object, then click-and-drag it to the desired location in the navigation pane on the left. You can move an object in your folder to the folder of another repository user.

To **restore** an object you deleted, double-click  (Trash). The object(s) you deleted appear in the right pane. Right-click on the object you want restored, and select **Restore** from the menu.

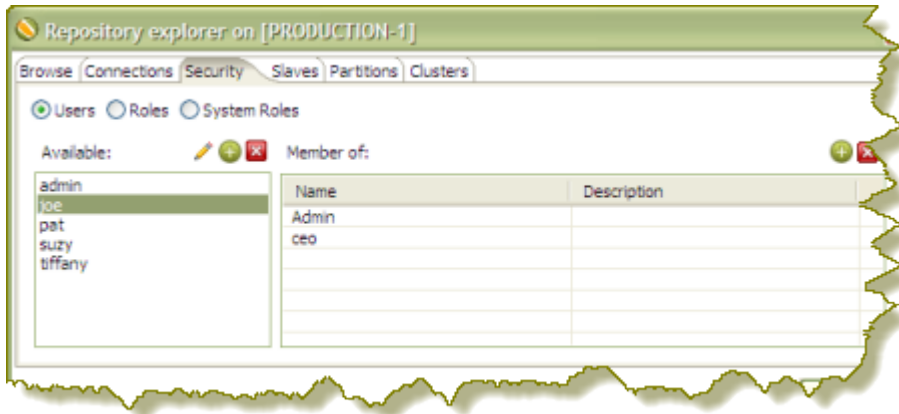
To **lock** a job or transformation from being edited by other users, select the job or transformation, right-click, and choose **Lock**. Enter a meaningful comment in the notes box that appears. A padlock icon appears next to jobs and transformation that have been locked. Locking and unlocking objects in the repository works like a toggle switch. When you release a lock on an object, the checkmark next to the Lock option disappears.

 **Note:** The lock status icons are updated on each PDI client only when the Repository Explorer is launched. If you want to refresh lock status in the Repository Explorer, exit and re-launch it.



In addition to managing content such as jobs and transformations, click the **Connections** tab to manage (create, edit, and delete) your database connections in the Enterprise Repository. See [Managing Connections](#) for more information about connecting to a database.

Click the **Security** tab to manage users and roles. Pentaho Data Integration comes with a default security provider. If you do not have an existing security such as LDAP or MSAD, you can use Pentaho Security to define users and roles. You must have administrative privileges to manage security. For more information, see the *Pentaho Data Integration Administrator's Guide*.



You can manage your slave servers (Data Integration and Carte instances) by clicking the **Slaves** tab. See [Setting Up a Slave Server](#) for instructions.

Click the **Partitions** and **Cluster** tabs to manage partitions and clusters. See [Creating a Cluster Schema](#) for more information.

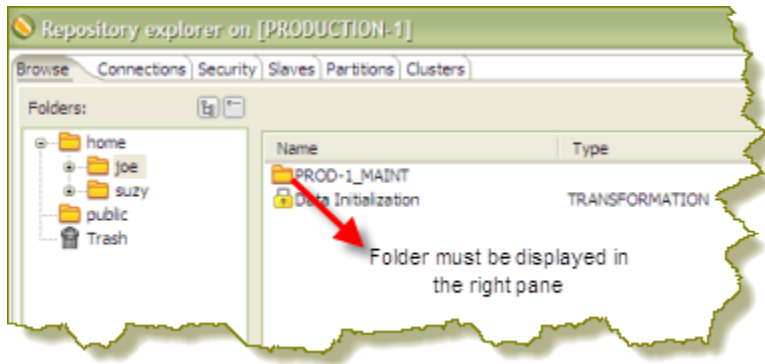
Setting Folder-Level Permissions

You can set permissions for jobs and transformations inside a folder in the repository. When you set permissions you determine which users have access to your content. You can set permissions by user or by role. You can also use the method described below to set permissions for individual objects in the repository. In that case, however, you'd select the individual job or transformation and set permissions.

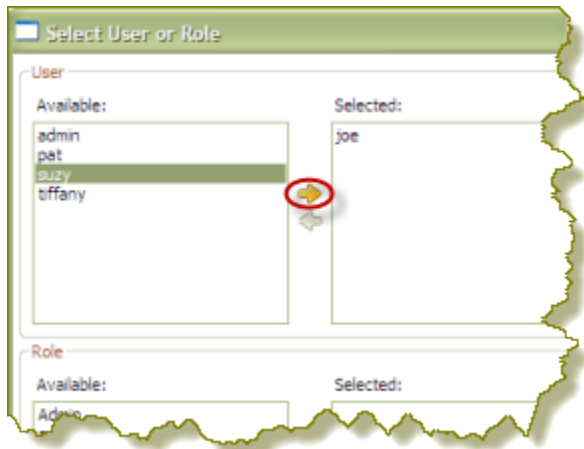
Follow the instructions below to set folder-level permissions.

1. Open the Repository Explorer (**Tools -> Repository -> Explore**).
2. Navigate to the folder to which you want permissions set and click to select it.

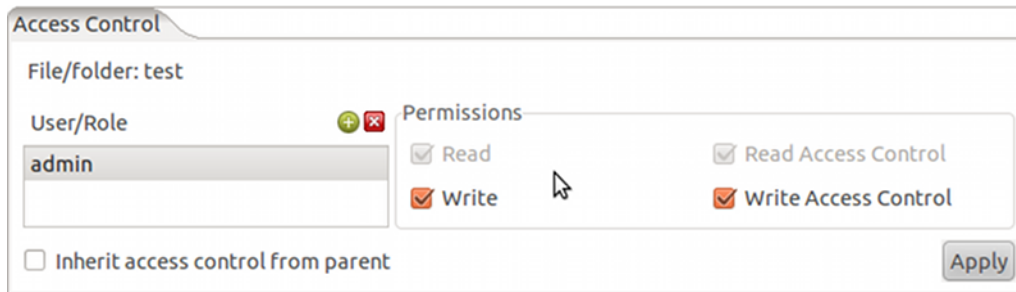
The folder must appear in the right pane before you can set permissions.



3. In the lower pane, under the **Permissions** tab, disable **Inherit security settings from parent**.
4. Click **+** (Add) to open the **Select User or Role** dialog box.
5. Select a user or role to add to the permission list. Use the yellow arrows to move the user or role in or out of the permissions list. Click **OK** when you are done.



6. In the lower pane, under the **Access Control** tab, enable the appropriate **Permissions** granted to your selected user or role.



If you change your mind, use **×** (Delete) to remove users or roles from the list.


7. Click **Apply** to apply permissions.

Working with Version Control

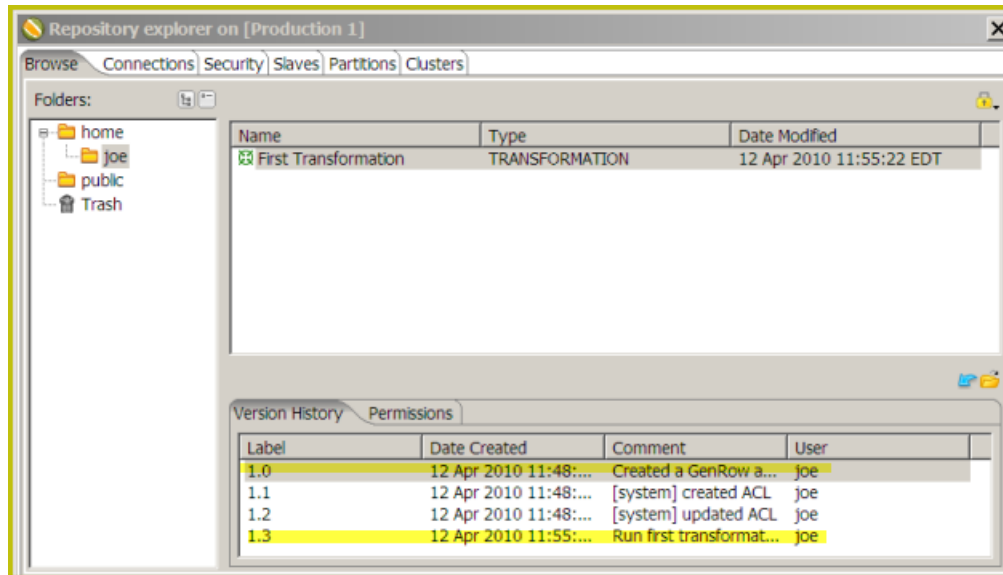
Whenever you save a job or transformation in the Enterprise Repository, you are prompted to provide a comment. Your comments are saved along with your job or transformation so that you can keep track of changes you make. If you have made a change to a transformation or job that you do not like, you can choose to restore a specific version of that job or transformation. It is important to provide descriptive version control comments, so that you can make good decisions when reverting to a version of a job or transformation.

Examining Revision History

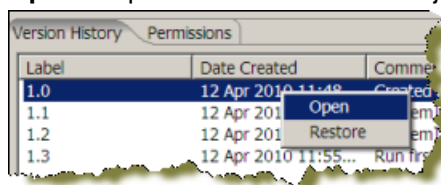
To examine revision history for a job or transformation...

1. In Spoon menubar, go to **Tools -> Repository -> Explore**. Alternatively click  in the Spoon menubar. The **Repository Explorer** window opens.
2. In the navigation pane on the left, locate and double-click the folder that contains your job or transformation. In the example below, there is one transformation listed in the folder called, "joe."
3. Click on a transformation or job from the list to select it. Notice that the **Version History** associated with transformation or job appears in the lower pane.

Administrative users see the **home** folders of all users on the system. If you are not an administrator you see your **home** and **public** folders. Your **home** folder is where you manage private content, such as transformations and jobs that are "in progress," for example. The **public** folder is where you store content that you want to share with others.



4. Right-click on the line under Version History that contains the transformation or job you want to examine. Choose **Open** to open the transformation or job in Spoon.



Restoring a Previously Saved Version of a Job or Transformation

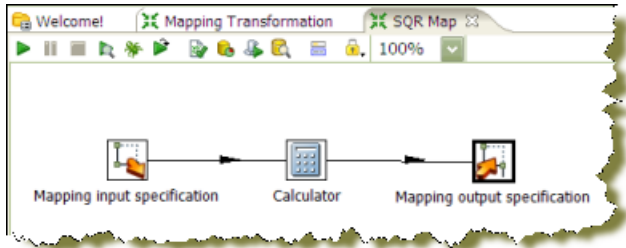
To restore a version of a job or transformation...


1. In Spoon menubar, go to **Tools -> Repository -> Explore**. The **Repository Explorer** window opens.
2. Browse through the folders to locate the transformation or job that has multiple versions associated with it.
3. Right-click on a transformation or job from the list to select it.
4. Select **Restore**.
5. Write a meaningful comment in the **Commit Comment** dialog box and click **OK**. The version is restored. Next time you open the transformation or job, the restored version is what you will see.

Reusing Transformation Flows with Mapping Steps




When you want to reuse a specific sequence of steps, you can turn the repetitive part into a *mapping*. A mapping is a standard transformation except that you can define mapping input and output steps as placeholders.

- Mapping Input Specification — the placeholder used for input from the parent transformation
- Mapping Output Specification — the placeholder from which the parent transformation reads data



 **Note:** Pentaho Data Integration samples that demonstrate the use of mapping steps are located at `...samples\mapping\Mapping`.

Below is the reference for the **Mapping (sub-transformation)** step:

| Option | Description |
|-------------------------------|---|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Mapping transformation | Specify the name of the mapping transformation file to execute at runtime. You can specify either a filename (XML/.ktr) or a transformation from the repository. The Edit button opens the specified transformation under a separate step in the Spoon Designer. |
| Parameters | Options under the Parameters tab allow you to define or pass PDI variables down to the mapping. This provides you with a high degree of customization. <ul style="list-style-type: none">  Note: It is possible to include variable expressions in the string values for the variable names.  Note: Important! Only those variables/values that are specified are passed down to the sub-transformation. |
| Input Tabs | Each of the Input tabs (may be missing) correspond to one Mapping Input Specification step in the mapping or sub-transformation. This means you can have multiple Input tabs in a single Mapping step. To add an Input tab, click Add Input . <ul style="list-style-type: none"> • Input source step name— The name of the step in the parent transformation (not the mapping) from which to read • Mapping target step name — The name of the step in the mapping (sub-transformation) to send the rows of data from the input source step • Is this the main data path? — Enable if you only have one input mapping ; you can leave the Mapping source step name and Output target step name fields blank • Ask these values to be renamed back on output? — Fields get renamed before they are transferred to the mapping transformation <ul style="list-style-type: none">  Note: Enabling this option renames the values back to their original names once they move |

| Option | Description |
|-------------------------------|---|
| | <p>to the Mapping output step. This option makes your sub-transformations more transparent and reusable.</p> <ul style="list-style-type: none"> • Step mapping description — Add a description of the mapping step • Source - mapping transformation mapping Enter the required field name changes |
| Output Tabs | <p>Each of the Output tabs (may be missing) correspond to one Mapping Output Specification step in the mapping or sub-transformation. This means you can have multiple Output tabs in a single Mapping step. To add an Output tab, click Add Output.</p> <ul style="list-style-type: none"> • Mapping source step — the name of the step in the mapping transformation (sub-transformation) where that will be read • Output target step name — the name of the step in the current transformation (parent) to send the data from the mapping transformation step to. • Is this the main data path? — Enable if you only have one output mapping and you can leave the Mapping source step and Output target step name fields above blank. • Step mapping description — Add a description to the output step mapping • Mapping transformation - target step field mapping — Enter the required field name changes |
| Add input / Add output | Add an input or output mapping for the specified sub-transformation |

Arguments, Parameters, and Variables

PDI has three paradigms for storing user input: arguments, parameters, and variables. Each is defined below, along with specific tips and configuration information.

Arguments

A PDI argument is a named, user-supplied, single-value input given as a command line argument (running a transformation or job manually from Pan or Kitchen, or as part of a script). Each transformation or job can have a maximum of 10 arguments. Each argument is declared as space-separated values given after the rest of the Pan or Kitchen line:

```
sh pan.sh -file:/example_transformations/example.ktr argOne argTwo argThree
```

In the above example, the values **argOne**, **argTwo**, and **argThree** are passed into the transformation, where they will be handled according to the way the transformation is designed. If it was not designed to handle arguments, nothing will happen. Typically these values would be numbers, words (strings), or variables (system or script variables, not PDI variables).

In Spoon, you can test argument handling by defining a set of arguments when you run a transformation or job. This is accomplished by typing in values in the **Arguments** fields in the **Execute a Job** or **Execute a Transformation** dialogue.

Parameters

Parameters are like local variables; they are reusable inputs that apply only to the specific transformation that they are defined in. When defining a parameter, you can assign it a default value to use in the event that one is not fetched for it. This feature makes it unique among dynamic input types in PDI.



Note: If there is a name collision between a parameter and a variable, the parameter will take precedence.

To define a parameter, right-click on the transformation workspace and select **Transformation settings** from the context menu (or just press **Ctrl-T**), then click on the **Parameters** tab.

Configuring SFTP VFS

To configure the connection settings for SFTP dialogues in PDI, you must create either variables or parameters for each relevant value. Possible values are determined by the VFS driver you are using.

You can also use parameters to substitute VFS connection details, then use them in the VFS dialogue where appropriate. For instance, these would be relevant credentials, assuming the parameters have been set:

```
sftp://${username}@${host}/${path}
```

This technique enables you to hide sensitive connection details, such as usernames and passwords.

See [VFS Properties](#) on page 41 for more information on VFS options. You can also see all of these techniques in practice in the **VFS Configuration Sample** sample transformation in the `/data-integration/samples/transformations/` directory.

VFS Properties


```
vfs . scheme . property . host
```

The **vfs** subpart is required to identify this as a virtual filesystem configuration property. The **scheme** subpart represents the VFS driver's scheme (or VFS type), such as `http`, `sftp`, or `zip`. The **property** subpart is the name of a VFS driver's ConfigBuilder's setter (the specific VFS element that you want to set). The **host** optionally defines a specific IP address or hostname that this setting applies to.

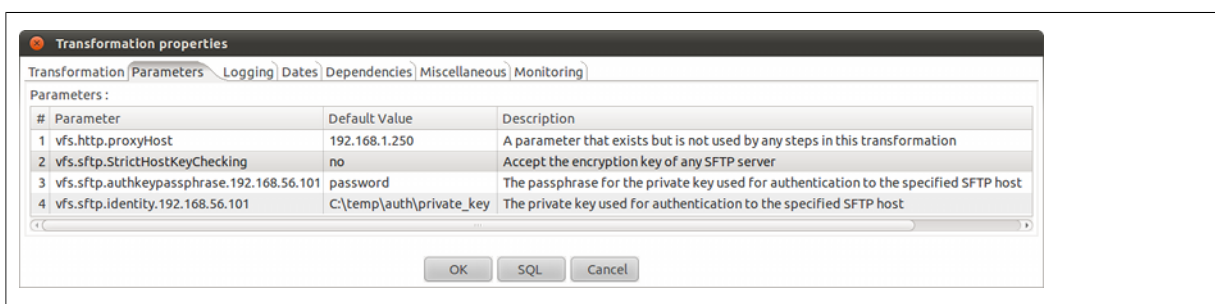
You must consult each scheme's API reference to determine which properties you can create variables for. Apache provides VFS scheme documentation at <http://commons.apache.org/vfs/apidocs/index.html>. The **org.apache.commons.vfs.provider** package lists each of the configurable VFS providers (`ftp`, `http`, `sftp`, etc.). Each provider has a **FileSystemConfigBuilder** class that in turn has **set*(FileSystemOptions, Object)** methods. If a

method's second parameter is a **String** or a **number** (Integer, Long, etc.) then you can create a PDI variable to set the value for VFS dialogues.

The table below explains VFS properties for the SFTP scheme. Each property must be declared as a PDI variable and preceded by the **vfs.sftp** prefix as defined above.

 **Note:** All of these properties are optional.


| SFTP VFS Property | Purpose |
|-----------------------|--|
| compression | Specifies whether zlib compression is used for the destination files. Possible values are zlib and none . |
| identity | The private key file (fully qualified local or remote path and filename) to use for host authentication. |
| authkeypassphrase | The passphrase for the private key specified by the identity property. |
| StrictHostKeyChecking | If this is set to no , the certificate of any remote host will be accepted. If set to yes , the remote host must exist in the known hosts file (~/.ssh/known_hosts). |



Variables


A variable in PDI is a piece of user-supplied information that can be used dynamically and programmatically in a variety of different scopes. A variable can be local to a single step, or be available to the entire JVM that PDI is running in.

PDI variables can be used in steps in both jobs and transformations. You define variables with the **Set Variable** step in a transformation, by hand through the **kettle.properties** file, or through the **Set Environment Variables** dialogue in the **Edit** menu.

The **Get Variable** step can explicitly retrieve a value from a variable, or you can use it in any PDI text field that has the diamond dollar sign  icon next to it by using a metadata string in either the Unix or Windows formats:

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

Both formats can be used and even mixed.

 **Note:** If there is a name collision with a parameter or argument, variables will defer.

Variable Scope

The scope of a variable is defined by the location of its definition. There are two types of variables: global environment variables, and Kettle variables. Both are defined below.

Environment Variables

This is the traditional variable type in PDI. You define an environment variable through the **Set Environment Variables** dialogue in the **Edit** menu, or by hand by passing it as an option to the Java Virtual Machine (JVM) with the **-D** flag.

Environment variables are an easy way to specify the location of temporary files in a platform-independent way; for example, the `${java.io.tmpdir}` variable points to the `/tmp/` directory on Unix/Linux/OS X and to the `C:\Documents and Settings\\Local Settings\Temp\` directory on Windows.

The only problem with using environment variables is that they cannot be used dynamically. For example, if you run two or more transformations or jobs at the same time on the same application server, you may get conflicts. Changes to the environment variables are visible to all software running on the virtual machine.

Kettle Variables

Kettle variables provide a way to store small pieces of information dynamically in a narrower scope than environment variables. A Kettle variable is local to Kettle, and can be scoped down to the job or transformation in which it is set, or up to a related job. The **Set Variable** step in a transformation allows you to specify the related job that you want to limit the scope to; for example, the parent job, grandparent job, or the root job.

Internal Variables

The following variables are always defined:

| Variable Name | Sample Value |
|--------------------------------------|---------------------|
| Internal.Kettle.Build.Date | 2010/05/22 18:01:39 |
| Internal.Kettle.Build.Version | 2045 |
| Internal.Kettle.Version | 4.1.0 |

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) |

 **Note:** In addition to the above, there are also **System** parameters, including command line arguments. These can be accessed using the [Get System Info](#) step in a transformation.

 **Note:** Additionally, you can specify values for variables in the **Execute a transformation** dialog box. If you include the variable names in your transformation they will appear in this dialog box.


Prototyping With Pentaho Data Integration

As of version 4.0, Pentaho Data Integration offers rapid prototyping of analysis schemas through a mix of processes and tools known as **Agile BI**. The Agile BI functions of Pentaho Data Integration are explained in this section, but there is no further instruction here regarding PDI installation, configuration, or use beyond ROLAP schema creation. If you need information related to PDI in general, consult the *Pentaho Data Integration Installation Guide* and/or the *Pentaho Data Integration User Guide* in the Pentaho Knowledge Base.

Creating a Prototype Schema With a Non-PDI Data Source

Your data sources must be configured, running, and available before you can proceed with this step.

Follow the below procedure to create a ROLAP schema prototype from an existing database, file, or data warehouse.

 **Note:** If you are already using PDI to create your data source, skip these instructions and refer to [Creating a Prototype Schema With a PDI Data Source](#) on page 44 instead.

1. Start Spoon and connect to your repository, if you are using one.

```
cd ~/pentaho/design-tools/data-integration/ && ./spoon.sh
```

2. Go to the **File** menu, then select the **New** sub-menu, then click on **Model**.

The interface will switch over to the **Model** perspective.

3. In the **Properties** pane on the right, click **Select**.

A data source selection window will appear.

4. Click the round green **+** icon in the upper right corner of the window.

The **Database Connection** dialogue will appear.

5. Enter in and select the connection details for your data source, then click **Test** to ensure that everything is correct. Click **OK** when you're done.

6. Select your newly-added data source, then click **OK**.

The **Database Explorer** will appear.

7. Traverse the database hierarchy until you get to the table you want to create a model for. Right-click the table, then select **Model** from the context menu.

The Database Explorer will close and bring you back to the Model perspective.

8. Drag items from the **Data** pane on the left and drop them into either the **Measures** or **Dimensions** groups in the **Model** pane in the center.

The Measures and Dimensions groups will expand to include the items you drag into them.

9. Select each new measure and dimension item, and modify its details accordingly in the **Properties** pane on the right.

10. Save your model through the **File** menu, or publish it to the BI Server using the **Publish** icon above the Model pane.

You now have a basic ROLAP schema. You should test it yourself before putting it into production. To do this, continue on to [Testing With Pentaho Analyzer and Report Wizard](#) on page 45.

Creating a Prototype Schema With a PDI Data Source

Context for the current task

1. Start Spoon and connect to your repository, if you are using one.

```
cd ~/pentaho/design-tools/data-integration/ && ./spoon.sh
```

2. Open the transformation that produces the data source you want to create a ROLAP schema for.

3. Right-click your output step, then select **Model** from the context menu.

4. Drag items from the **Data** pane on the left and drop them into either the **Measures** or **Dimensions** groups in the **Model** pane in the center.

The Measures and Dimensions groups will expand to include the items you drag into them.

5. Select each new measure and dimension item, and modify its details accordingly in the **Properties** pane on the right.
6. Save your model through the **File** menu, or publish it to the BI Server using the **Publish** icon above the Model pane.

You now have a basic ROLAP schema. You should test it yourself before putting it into production. To do this, continue on to [Testing With Pentaho Analyzer and Report Wizard](#) on page 45.

Testing With Pentaho Analyzer and Report Wizard

You must have an analysis schema with at least one measure and one dimension, and it must be currently open and focused on the Model perspective in Spoon.

This section explains how to use the embedded Analyzer and Report Design Wizard to test a prototype analysis schema.

1. While in the Model perspective, select your visualization method from the drop-down box above the Data pane (it has a **New:** to its left), then click **Go**.

The two possible choices are: **Pentaho Analyzer** and **Report Wizard**. You do not need to have license keys for Pentaho Analysis or Pentaho Reporting in order to use these preview tools.

2. Either the Report Design Wizard will launch in a new sub-window, or Pentaho Analyzer will launch in a new tab. Use it as you would in Report Designer or the Pentaho User Console.

3. When you have explored your new schema, return to the Model perspective by clicking **Model** in the upper right corner of the Spoon toolbar, where all of the perspective buttons are.

Do not close the tab; this will close the file, and you will have to reopen it in order to adjust your schema.

4. If you continue to refine your schema in the Model perspective, you must click the **Go** button again each time you want to view it in Analyzer or Report Wizard; the Visualize perspective does not automatically update according to the changes you make in the Modeler.

You now have a preview of what your model will look like in production. Continue to refine it through the Model perspective, and test it through the Visualize perspective, until you meet your initial requirements.

Prototypes in Production

Once you're ready to move your analysis schema to production, use the **Publish** button above the Model pane in the Model perspective, and use it to connect to your production BI Server.

You can continue to refine your schema if you like, but it must be republished each time you want to redeploy it.

Managing Connections

Pentaho Data Integration allows you to define connections to multiple databases provided by multiple database vendors (MySQL, Oracle, Postgres, and many more). Pentaho Data Integration ships with the most suitable JDBC drivers for supported databases and its primary interface to databases is through JDBC. Vendors write a driver that matches the JDBC specification and Pentaho Data Integration uses the driver. Unless you require extensive debugging or have other needs, you won't ever need to write your own database driver.

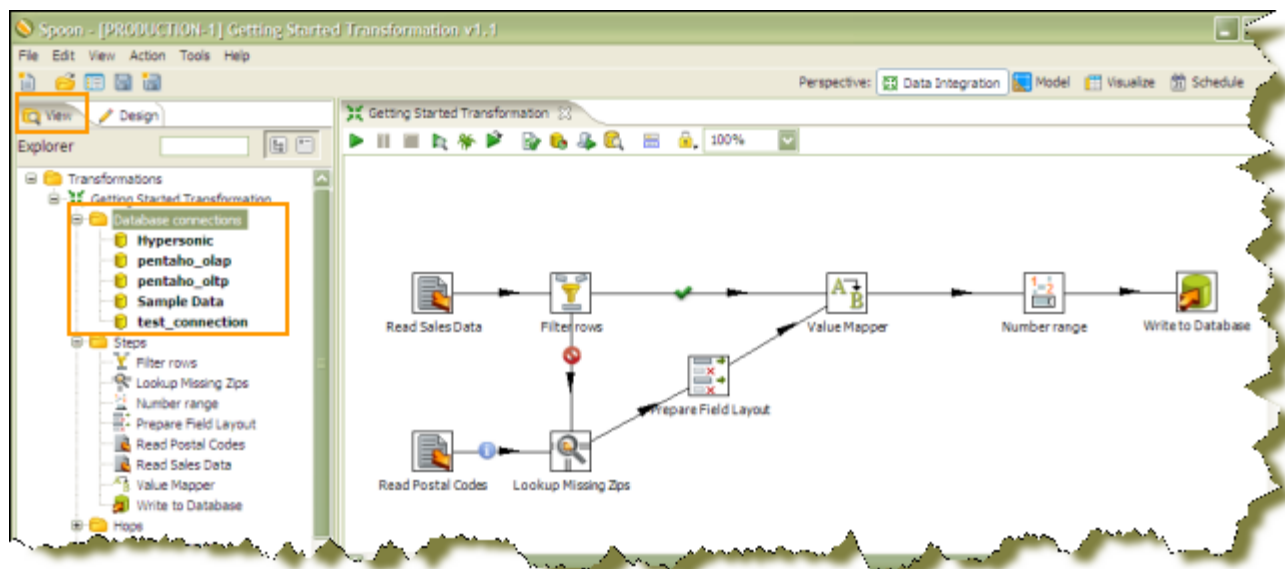
Note:



Pentaho recommends that you avoid using ODBC connections. The ODBC to JDBC bridge driver does not always provide exact match and adds another level of complexity that may affect performance. The only time you may have to use ODBC is if there is no available JDBC driver.

When you define a database connection, the connection information (user name, password, port number, and so on) is stored in the Pentaho Enterprise Repository and is available to other users when they connect to the repository. If you are not using the Pentaho Enterprise Repository, the database connection information is stored in the XML file associated with a transformation or job.

Connections that are available for use with a transformation or job are listed under the **Database Connection** node in the explorer **View** in Spoon.



There are several ways to define a new database connection:

- In Spoon, go to **File -> New -> Database Connection**.
- In Spoon, under **View**, right-click **Database connections** and choose **New**.
- In Spoon, under **View**, right-click **Database connections** and choose **New Connection Wizard**.

Adding a JDBC Driver

Before you can connect to a data source in any Pentaho server or client tool, you must first install the appropriate database driver. Your database administrator, CIO, or IT manager should be able to provide you with the proper driver JAR. If not, you can download a JDBC driver JAR file from your database vendor or driver developer's Web site. Once you have the JAR, follow the instructions below to copy it to the driver directories for all of the BI Suite components that need to connect to this data source.



Note: Microsoft SQL Server users frequently use an alternative, non-vendor-supported driver called JTDS. If you are adding an MSSQL data source, ensure that you are installing the correct driver.


Backing up old drivers

You must also ensure that there are no other versions of the same vendor's JDBC driver installed in these directories. If there are, you may have to back them up and remove them to avoid confusion and potential class loading problems.


This is of particular concern when you are installing a driver JAR for a data source that is the same database type as your Pentaho solution repository. If you have any doubts as to how to proceed, contact your Pentaho support representative for guidance.

Installing JDBC drivers

Copy the driver JAR file to the following directories, depending on which servers and client tools you are using (Dashboard Designer, ad hoc reporting, and Analyzer are all part of the BI Server):

 **Note: For the BI Server:** before copying a new JDBC driver, ensure that there is not a different version of the same JAR in the destination directory. If there is, you must remove the old JAR to avoid version conflicts.

- **BI Server:** /pentaho/server/biserver-ee/tomcat/lib/
- **Enterprise Console:** /pentaho/server/enterprise-console/jdbc/
- **Data Integration Server:** /pentaho/server/data-integration-server/tomcat/webapps/pentaho-di/WEB-INF/lib/
- **Data Integration client:** /pentaho/design-tools/data-integration/libext/JDBC/
- **Report Designer:** /pentaho/design-tools/report-designer/lib/jdbc/
- **Schema Workbench:** /pentaho/design-tools/schema-workbench/drivers/
- **Aggregation Designer:** /pentaho/design-tools/agg-designer/drivers/
- **Metadata Editor:** /pentaho/design-tools/metadata-editor/libext/JDBC/

 **Note:** To establish a data source in the Pentaho Enterprise Console, you must install the driver in both the Enterprise Console and the BI Server or Data Integration Server. If you are just adding a data source through the Pentaho User Console, you do not need to install the driver to Enterprise Console.

Restarting


Once the driver JAR is in place, you must restart the server or client tool that you added it to.

Connecting to a Microsoft SQL Server using Integrated or Windows Authentication

The JDBC driver supports Type 2 integrated authentication on Windows operating systems through the **integratedSecurity** connection string property. To use integrated authentication, copy the **sqljdbc_auth.dll** file to all the directories to which you copied the JDBC files.


The **sqljdbc_auth.dll** files are installed in the following location:

```
<installation directory>\sqljdbc_<version>\<language>\auth\
```


 **Note:** Use the **sqljdbc_auth.dll** file, in the x86 folder, if you are running a 32-bit Java Virtual Machine (JVM) even if the operating system is version x64. Use the **sqljdbc_auth.dll** file in the x64 folder, if you are running a 64-bit JVM on a x64 processor. Use the **sqljdbc_auth.dll** file in the IA64 folder, you are running a 64-bit JVM on an Itanium processor.

Defining Database Connections

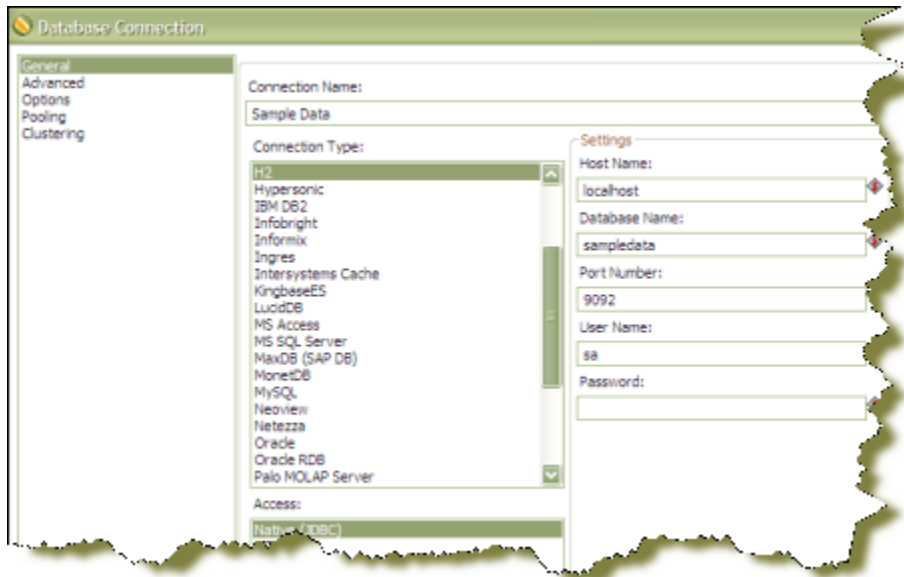
You must have information about your database, such as your database type, port number, user name and password, before you define a database connection. You can also set connection properties using variables. Variables (🔑) provide you with the ability to access data from multiple database types using the same transformations and jobs.

 **Note:** Make sure to use clean ANSI SQL that works on all used database types in the latter case.

1. Right-click the **Database connections** in the tree and choose **New** or **New Connection Wizard**.

 **Note:** Alternatively, double-click **Database Connections**, or select **Database Connections** and press <F3>. The connection wizard allows you to define a connection quickly. If you require advanced features like pooling and clustering, don't use the connection wizard.

Depending on your choice, the **Database Connection** dialog box (or wizard) appears. The wizard requires that you provide the same type of basic information as required in the Database Connection dialog box shown below:



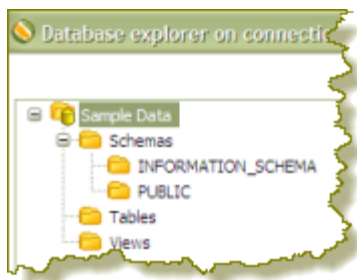
2. In the **Connection Name** field, enter a name that uniquely identifies your new connection.
3. Under **Connection Type**, select the database you are connecting to (for example, MySQL, Oracle, and so on).
4. Under **Access**, select your method of access. This will be **Native (JDBC)**, **ODBC**, or **JNDI**. Available access types depend on the type of database to which you are connecting.
5. Under **Settings**, in the **Host Name** field, enter the name of the server that hosts the database to which you are connecting. Alternatively, you can specify the host by IP address.
6. In the **Database Name** field, enter the name of the database to which you are connecting. If you are using a ODBC connection, enter the Data Source Name (DSN) in this field.
7. In the **Port Number** field, enter the TCP/IP port number if it is different from the default.
8. Optionally, enter the **User name** used to connect to the database.
9. Optionally, enter the **Password** used to connect to the database.

10. Click **Test**.

A confirmation message displays if Spoon is able to establish a connection with the target database.

11. Click **OK** to save your entries and exit the Database Connection dialog box.

Click **Explore** to open the **Database Explorer** for an existing connection, or right-click on the connection in View mode and select **Explore**.



Click **Feature List** to expose the JDBC URL, class, and various database settings for the connection such as the list of reserved words.

| # | Parameter | Value |
|----|--|-----------|
| 1 | Database type | H2 |
| 2 | Access type | Native |
| 3 | Database name | sampled |
| 4 | Server hostname | localhost |
| 5 | Service port | 9092 |
| 6 | Username | sa |
| 7 | Informix server name | |
| 8 | Extra attribute [USE_POOLING] | N |
| 9 | Extra attribute [IS_CLUSTERED] | N |
| 10 | Extra attribute [SUPPORTS_BOOLEAN_DATA_TYPE] | |

Working with JNDI Connections

If you are developing transformations and jobs that will be deployed on an application server such as the BI platform running on JBoss, you can configure your database connections using JNDI. Pentaho has supplied a way of configuring a JNDI connection for "local" Pentaho Data Integration use so that you do not have an application server continuously running during the development and testing of transformations. To configure, edit the properties file called **jdbc.properties** located at `... \data-integration-server \pentaho-solutions \system \simple-jndi`.



Note: It is important that the information stored in **jdbc.properties** mirrors the content of your application server data sources.

Working with JNDI Connections in Carte and Spoon

The Carte server and Spoon use JNDI in the same way. The information below may be helpful in instances when you have a transformation that uses a JNDI data source connection and that runs on a remote Carte server. There are three ways to set the location of the `Simple-JNDI jdbc.properties` location:

1. Add connections to the `<pdj-install>/simple-jndi/jdbc.properties` file. This is the default location for Carte.
2. Modify the **carte.bat** to execute with the command line option: `org.osjava.sj.root=<simple-jndi-path>`.
3. Modify the **carte.bat** to execute with the command line option `KETTLE_JNDI_ROOT=<simple-jndi-path>`.

Database-Specific Options

Options in the Database Connection dialog box allow you to set database-specific options for the connection by adding parameters to the generated URL.

Adding Database-Specific Options

Follow the instructions below to add parameters associated with **Options** in the **Database Connections** dialog box:

1. Select the next available row in the parameter table.
2. Enter a valid parameter name and its corresponding value.

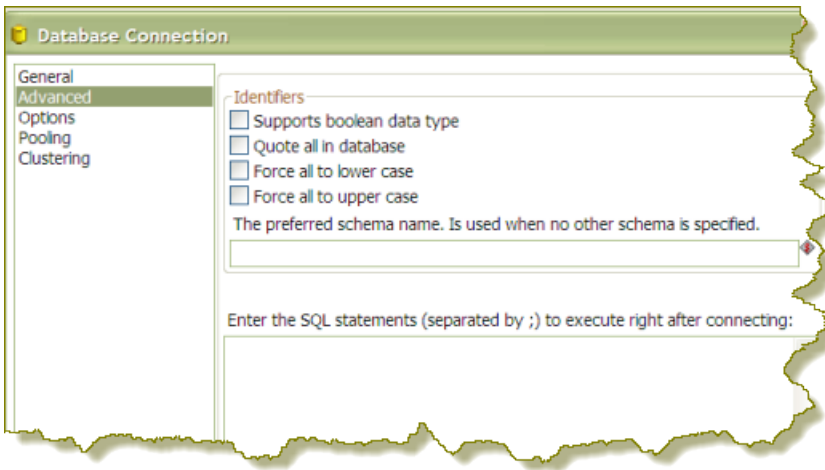


Note: For more database-specific configuration help, click **Help**. A new browser opens and displays additional information about configuring the JDBC connection for the currently selected database type.

3. Click **OK** to save your entries.

Database Connections Advanced Configurations

The **Advanced** option in the Database Connection dialog box allows you to configure properties that are, for most part, associated with how SQL is generated. These options allow you to set a standard across all of your SQL tools, ETL tools and design tools. All database table names and column names are always upper case or lower case no matter what users do in the tools.



| Feature | Description |
|------------------------------------|---|
| Supports boolean data types | Enable to instruct Pentaho Data Integration to use native boolean data types if supported by the database. |
| Quote all in database | Enable to instruct the databases to use a case-sensitive tablename; (for example MySQL is case-sensitive on Linux but not case sensitive on Windows. If you quote the identifiers, the databases will use a case sensitive tablename. |
| Force all to lower case | Enable to force all identifiers to lower case. |
| Force all to upper case | Enable to force all identifiers to upper case. |
| Preferred schema name... | Enter the preferred schema name to use, (for example, MYSCHEMA). |
| Enter SQL name... | Enter the SQL statement used to initialize a connection. |

More About Quoting

Pentaho has implemented a database-specific quoting system that allows you to use any name or character acceptable to the supported databases' naming conventions.

Pentaho Data Integration contains a list of reserved words for most of the supported databases. To ensure that quoting behaves correctly, Pentaho has implemented a strict separation between the schema (user/owner) of a table and the table name itself. Doing otherwise makes it impossible to quote tables or fields with one or more periods in them correctly. Placing periods in table and field names is common practice in some ERP systems (for example, fields such as "V.A.T.")

To avoid quoting-related errors, a rule stops the Pentaho Data Integration from performing quoting activity when there is a start or end quote in the table name or schema. This allows you to specify the quoting mechanism yourself.

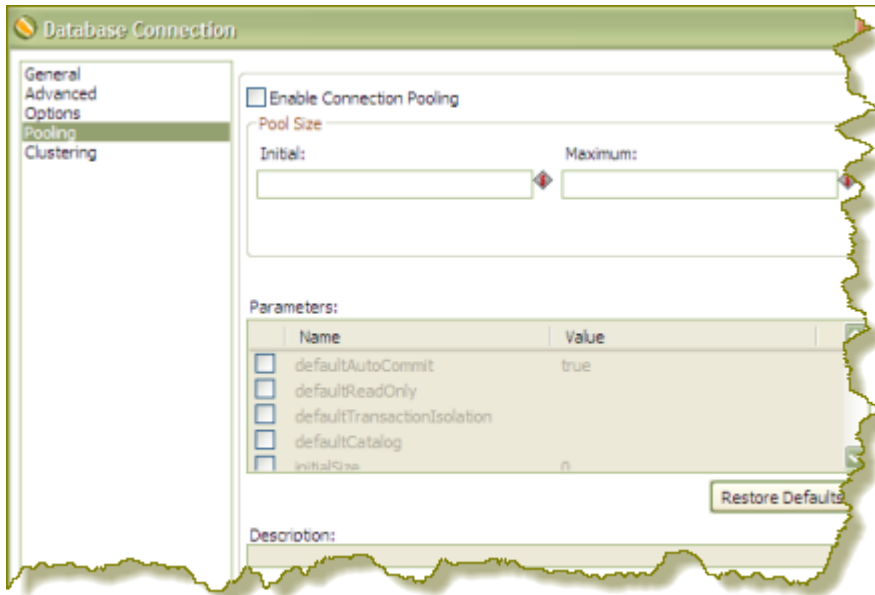
Connection Pooling

Instead of having a connection open for each individual step in a transformation, you can set up a connection *pool* and define options like the initial pool size, maximum pool size, and connection pool parameters. For example, you might start with a pool of ten or fifteen connections, and as you run jobs or transformations, the unused connections drop off. Pooling helps control database access, especially if you have transformations that contain many steps and that require a large number of connections. Pooling can also be implemented when your database licencing restricts the number of active concurrent connections.

The table below provides a more detailed description of the pooling options available:

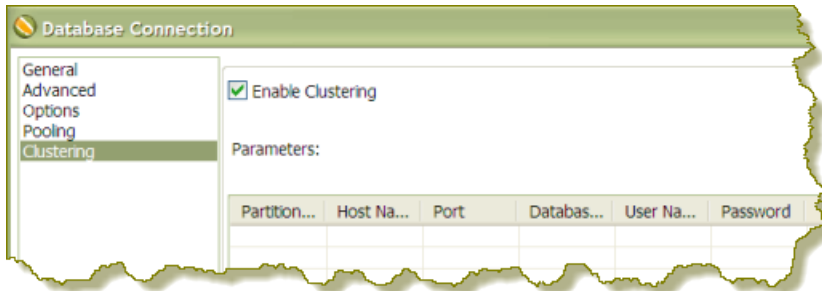
| Feature | Description |
|----------------------------------|---|
| Enable connection pooling | Enables connection pooling |
| Pool Size | Sets the <i>initial</i> size of the connection pool; sets the <i>maximum</i> number of connections in the connection pool |
| Parameters | Allows you to define additional custom pool parameters; click Restore Defaults when appropriate |

| Feature | Description |
|-------------|---|
| Description | Allows you to add a description for your parameters |



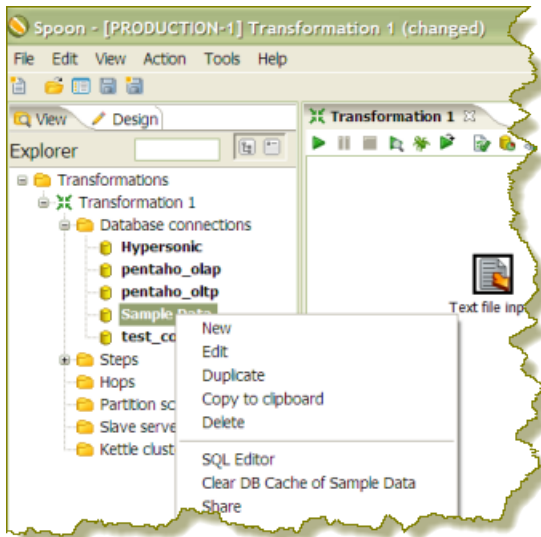
Clustering

This option 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 **Host Name, Port, Database, User Name, and Password** for connecting to the partition.



Editing, Duplicating, Copying, and Deleting Connections

The table below contains information about other database-related connection tasks you can perform. Refer to the image as you read the task descriptions.



| Task | Description |
|---------------------------------|--|
| Edit a Connection | Right-click on the connection name and select Edit . |
| Duplicate a Connection | Right-click on the connection name and select Duplicate . |
| Copy to a Clipboard | Allows you to copy the XML defining the step to the clipboard. You can then paste this step into another transformation. Double-click on the connection name in the tree or right-click on the connection name and select Copy to Clipboard . |
| Delete a Connection | Double-click on the connection name in the tree or right-click on the connection name and select Delete . |
| SQL Editor | To execute SQL command against an existing connection, right-click on the connection name and select SQL Editor . |
| Clear the Database Cache | To speed up connections Pentaho Data Integration uses a database cache. When the information in the cache no longer represents the layout of the database, right-click on the connection in the tree and select Clear DB Cache.... This command is commonly used when databases tables have been changed, created or deleted. |
| Share a Connection | Rather than redefining a connection each time you create a job or transformation on your <i>local device</i> , right-click and select Share to share the connection information among jobs and transformations. |
| Exploring the Database | Double-click on the connection name in the tree or right-click on the connection name and select Explore . |
| Show dependencies | Right-click a connection name and select Show dependencies to see all of the transformations and jobs that use this database connection. |

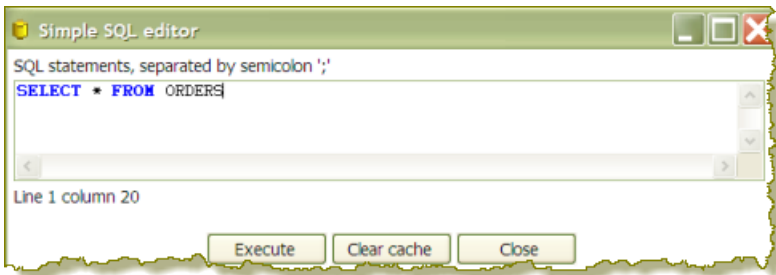
Using the SQL Editor

The **SQL Editor** is good tool to use when you must execute standard SQL commands for tasks such as creating tables, dropping indexes and modifying fields. The SQL Editor is used to preview and execute DDL (Data Definition Language) generated by Spoon such as "create/alter table," "create index," and "create sequence" SQL commands. For example, if you add a Table Output step to a transformation and click the SQL button at the bottom of the Table Input dialog box, Spoon automatically generates the necessary DDL for the output step to function properly and presents it to the end user through the SQL Editor.

Below are some points to consider:

- Multiple SQL Statements must be separated by semi-colons.

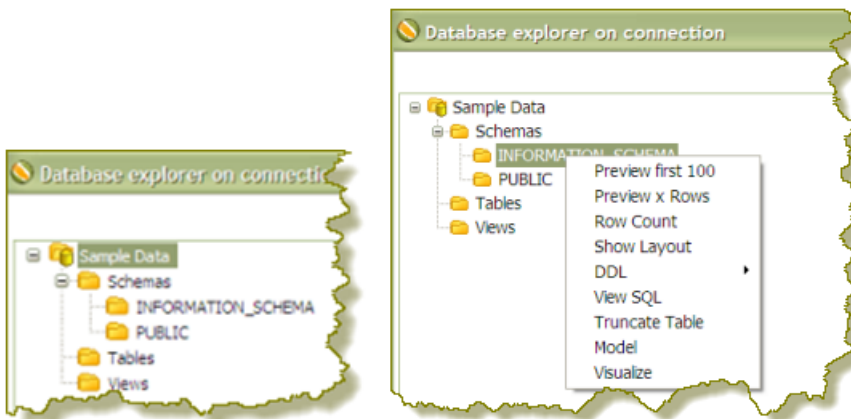
- Before SQL Statements are sent to the database to be executed, Spoon removes returns, line-feeds, and separating semi-colons.
- Pentaho Data Integration clears the database cache for the database connection on which you launch DDL statements.




The SQL Editor does not recognize the dialects of all supported databases. That means that creating stored procedures, triggers, and other database-specific objects may pose problems. Consider using the tools that came with the database in these instances.

Using the Database Explorer

The **Database Explorer** allow you to explore configured database connections. The Database Explorer also supports tables, views, and synonyms along with the catalog, schema, or both to which the table belongs.



A right-click on the selected table provides quick access to the following features:

| Feature | Description |
|--------------------------|---|
| Preview first 100 | Returns the first 100 rows from the selected table |
| Preview x Rows | Prompts you for the number of rows to return from the selected table |
| Row Count | Specifies the total number of rows in the selected table |
| Show Layout | Displays a list of column names, data types, and so on from the selected table |
| DDL | Generates the DDL to create the selected table based on the current connection type; the drop-down |
| View SQL | Launches the Simple SQL Editor for the selected table |
| Truncate Table | Generates a TRUNCATE table statement for the current table  Note: The statement is commented out by default to prevent users from accidentally deleting the table data |

Unsupported Databases

Contact Pentaho if you want to access a database type that is not yet supported. A few database types are not supported in this release due to the lack of a sample database, software, or both. It is generally possible to read from unsupported databases by using the generic database driver through an ODBC or JDBC connection.

You can add or replace a database driver files in the **libext** directory located under **...\design-tools\data-integration**.

Performance Monitoring and Logging

Pentaho Data Integration provides you with several methods in which to monitor the performance of jobs and transformations. Logging offers you summarized information regarding a job or transformation such as the number of records inserted and the total elapsed time spent in a transformation. In addition, logging provides detailed information about exceptions, errors, and debugging details.

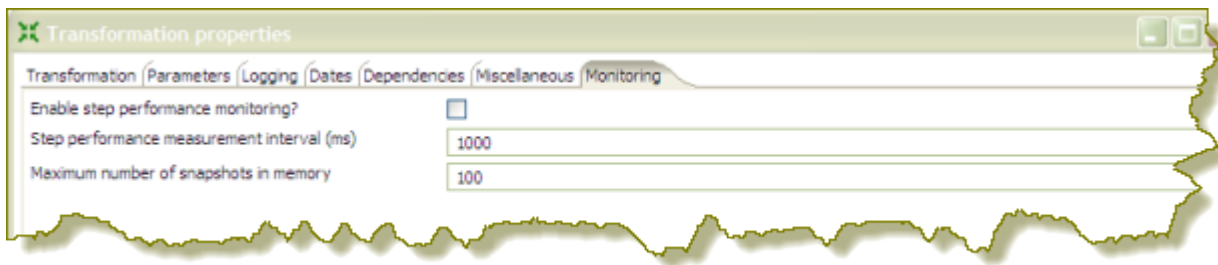
Reasons you may want to enable logging and step performance monitoring include: determining if a job completed with errors or to review errors that were encountered during processing. In headless environments, most ETL in production is not run from the graphical user interface and you need a place to watch initiated job results. Finally, performance monitoring provides you with useful information for both current performance problems and capacity planning.

If you are an administrative user and want to monitor jobs and transformations in the Pentaho Enterprise Console, you must first set up logging and performance monitoring in Spoon. For more information about monitoring jobs and transformation in the Pentaho Enterprise Console, see the *Pentaho Data Integration Administrator's Guide*.

Monitoring Step Performance

Pentaho Data Integration provides you with a tool for tracking the performance of individual steps in a transformation. By helping you identify the slowest step in the transformation, you can fine-tune and enhance the performance of your transformations.

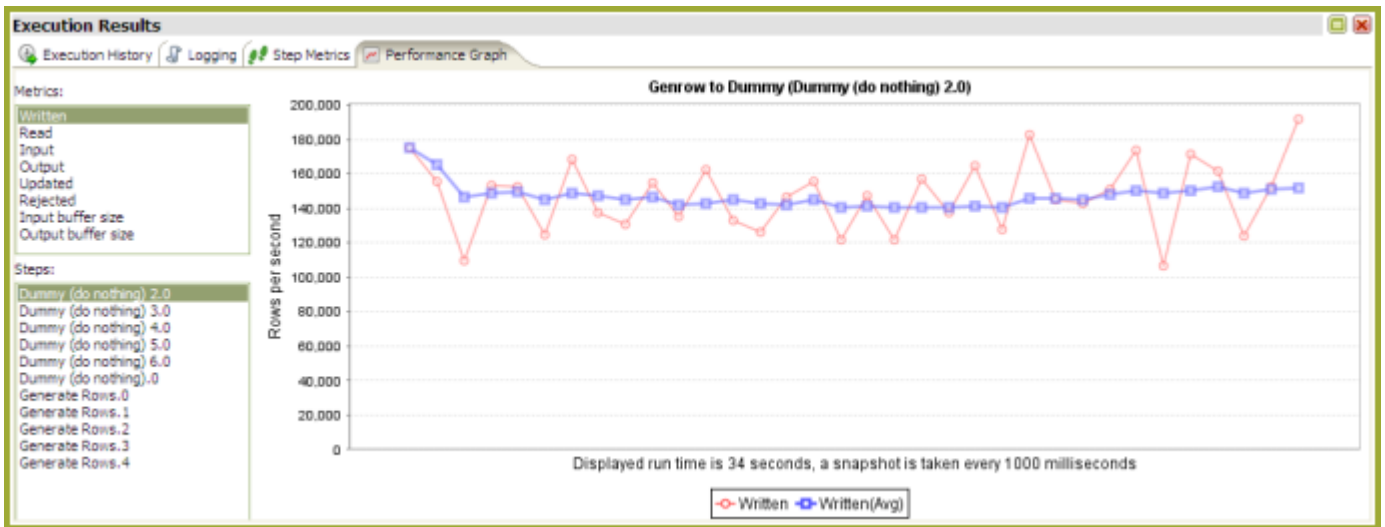
You enable the step performance monitoring in the **Transformation Properties** dialog box. To access the dialog box right-click in the workspace that is displaying your transformation and choose, **Transformation Settings**. You can also access this dialog box, by pressing **<CTRL + T>**.



As shown in the sample screen capture above, the option to track performance (**Enable step performance monitoring?**) is not selected by default. Step performance monitoring may cause memory consumption problems in long-running transformations. By default, a performance snapshot is taken for all the running steps every second. This is not a CPU-intensive operation and, in most instances, does not negatively impact performance unless you have many steps in a transformation or you take a lot of snapshots (several per second, for example). You can control the number of snapshots in memory by changing the default value next to **Maximum number of snapshots in memory**. In addition, if you run in Spoon locally you may consume a fair amount of CPU power when you update the JFreeChart graphics under the Performance tab. Running in "headless" mode (Kitchen, Pan, DI Server (slave server), Carte, Pentaho BI platform, and so on) does not have this drawback and should provide you with accurate performance statistics.

Using Performance Graphs

If you configured step performance monitoring, with database logging (optional), you can view the performance evolution graphs. Performance graphs provide you with a visual interpretation of how your transformation is processing.




Follow the instructions below to set up a performance graph history for your transformation.

1. Right-click in the workspace (canvas) where you have an open transformation. Alternatively, press <CTRL +T>. The **Transformation Properties** dialog box appears.
2. In the Transformation Properties dialog box, click the **Logging** tab. Make sure **Performance** is selected in the navigation pane on the left.
3. Under **Logging** enter the following information:

| Option | Description |
|-------------------------------------|--|
| Log Connection | Specifies the database connection you are using for logging; you can configure a new connection by clicking New . |
| Log Table Schema | Specifies the schema name, if supported by your database |
| Log Table Name | Specifies the name of the log table (for example L_ETL) |
| Logging interval (seconds) | Specifies the interval in which logs are written to the table |
| Log record timeout (in days) | Specifies the number of days old log entries in the table will be kept before they are deleted |

4. Enable the fields you want to log or keep the defaults.
5. Click **SQL** to create your log table.
The Simple SQL Editor appears.
6. Click **Execute** to execute the SQL code for your log table, then click **OK** to exit the **Results** dialog box.

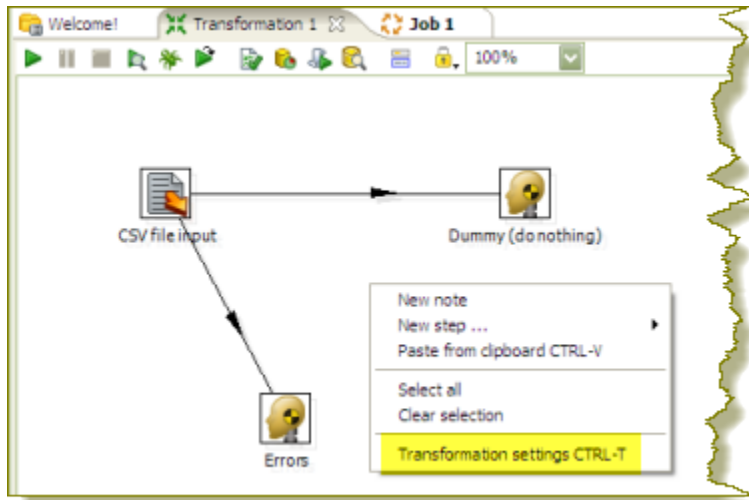
 **Note:** You *must* execute the SQL code to create the log table.

7. Click **Close** to exit the Simple SQL Editor.
8. Click **OK** to exit the Transformation Properties dialog box.

Logging Steps

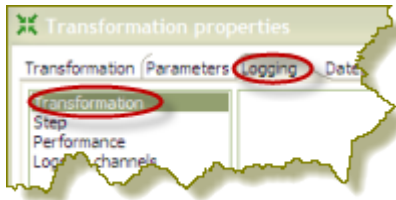
Follow the instructions below to create a log table that keeps history of step-related information associated with your transformation.

1. Right-click in the workspace (canvas) where you have an open transformation. Alternatively, press <CTRL +T>.



The **Transformation Properties** dialog box appears.

- In the Transformation Properties dialog box, click the **Logging** tab. Make sure **Step** is selected in the navigation pane on the left.




d

- Under **Logging** enter the following information:

| Option | Description |
|-------------------------------------|--|
| Log Connection | Specifies the database connection you are using for logging; you can configure a new connection by clicking New . |
| Log Table Schema | Specifies the schema name, if supported by your database |
| Log Table Name | Specifies the name of the log table (for example L_STEP) |
| Logging interval (seconds) | Specifies the interval in which logs are written to the table |
| Log record timeout (in days) | Specifies the number of days old log entries in the table will be kept before they are deleted |

- Enable the fields you want to log or keep the defaults.
- Click **SQL** to create your log table.
The Simple SQL Editor appears.
- Click **Execute** to execute the SQL code for your log table, then click **OK** to exit the **Results** dialog box.

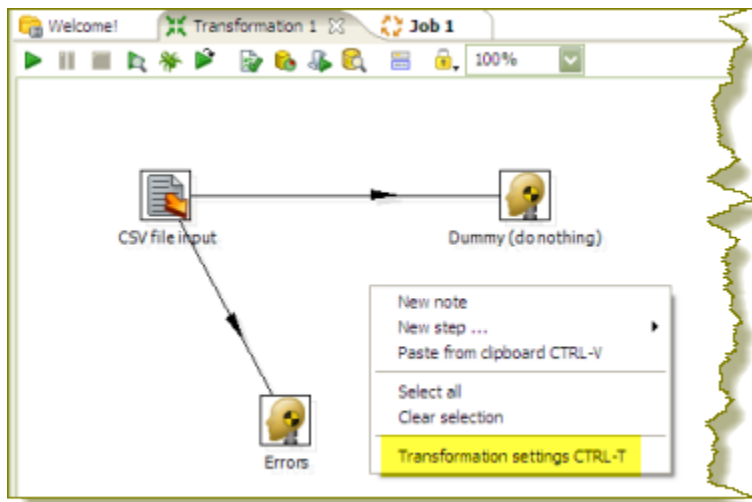
 **Note:** You *must* execute the SQL code to create the log table.

- Click **Close** to exit the Simple SQL Editor.
- Click **OK** to exit the Transformation Properties dialog box.

Logging Transformations

Follow the instructions below to create a log table for transformation-related processes:

- Right-click in the workspace (canvas) where you have an open transformation. Alternatively, press <CTRL +T>.



The **Transformation Properties** dialog box appears.


- In the Transformation Properties dialog box, click the **Logging** tab. Make sure **Transformation** is selected in the navigation pane on the left.



- Under **Logging** enter the following information:

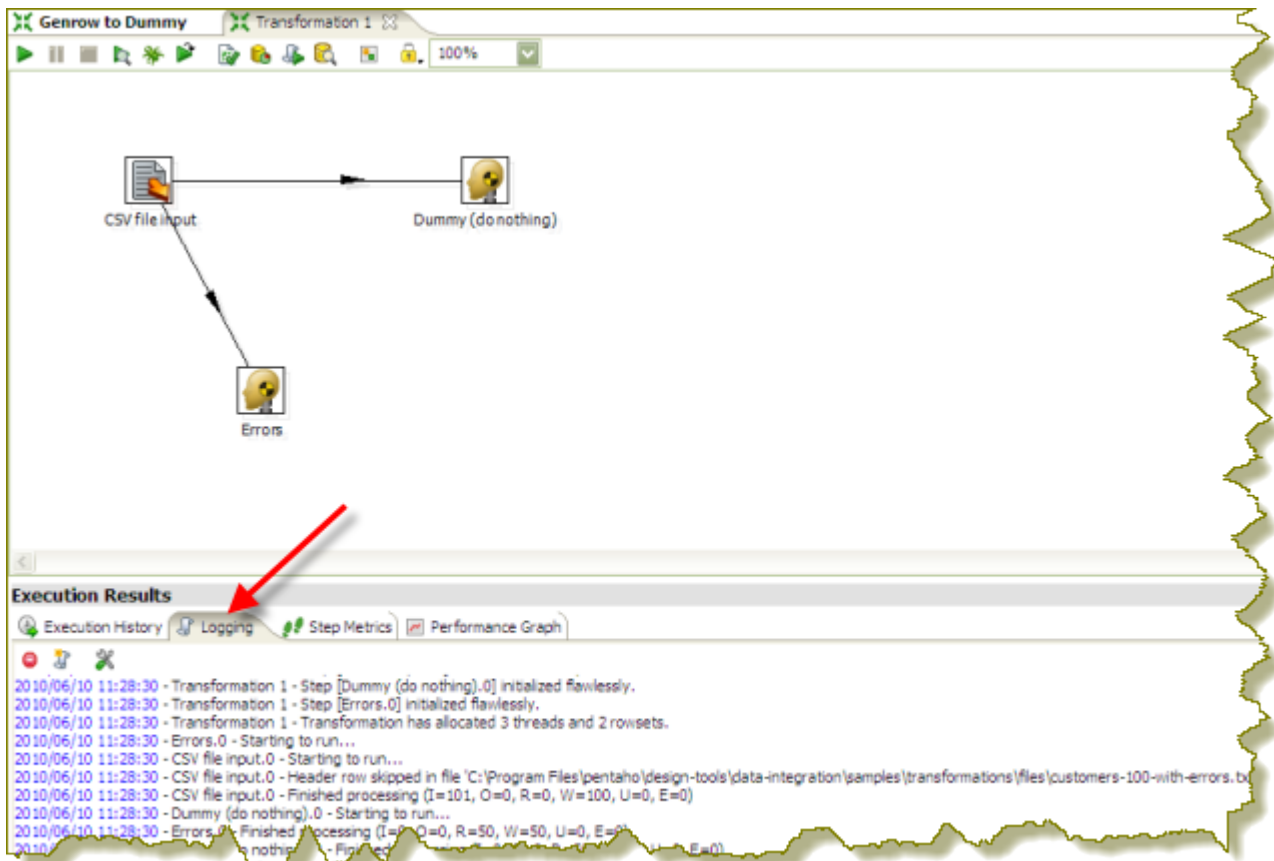
| Option | Description |
|-------------------------------------|---|
| Log Connection | Specifies the database connection you are using for logging; you can configure a new connection by clicking New . |
| Log Table Schema | Specifies the schema name, if supported by your database |
| Log Table Name | Specifies the name of the log table (for example L_ETL) |
| Logging interval (seconds) | Specifies the interval in which logs are written to the table |
| Log record timeout (in days) | Specifies the number of days old log entries in the table will be kept before they are deleted |
| Log size limit in lines | Limits the number of lines that are stored in the LOG_FIELD (when selected under Fields to Log); when the LOG_FIELD is enabled Pentaho Data Integration will store logging associated with the transformation in a long text field (CLOB) |

- Enable the fields you want to log or keep the defaults.
- Click **SQL** to create your log table.
The Simple SQL Editor appears.
- Click **Execute** to execute the SQL code for your log table, then click **OK** to exit the **Results** dialog box.

 **Note:** You *must* execute the SQL code to create the log table.

- Click **Close** to exit the Simple SQL Editor.
- Click **OK** to exit the Transformation Properties dialog box.

The next time you run your transformation, logging information will be displayed under the **Logging** tab. If you run your transformation on a slave server, and are an administrative user, you can view log information on the Pentaho Enterprise Console. See the *Pentaho Data Integration Administrator's Guide* for details.




 **Note:** If the log does not appear, click  (Log Settings) to adjust your logging parameters.

Pentaho Data Integration Performance Tuning Tips

The tips described here may help you to identify and correct performance-related issues associated with PDI transformations.


| Step | Tip | Description |
|------|-----------------------------|--|
| JS | Turn off compatibility mode | <p>Rewriting JavaScript to use a format that is not compatible with previous versions is, in most instances, easy to do and makes scripts easier to work with and to read. By default, old JavaScript programs run in compatibility mode. That means that the step will process like it did in a previous version. You may see a small performance drop because of the overload associated with forcing compatibility mode and change the code as shown below:</p> <ul style="list-style-type: none"> • <code>intField.getInteger() --> intField</code> • <code>numberField.getNumber() --> numberField</code> • <code>dateField.getDate() --> dateField</code> • <code>bigNumberField.getBigNumber() --> bigNumberField</code> • and so on... <p>Instead of Java methods, use the built-in library. Notice that the resulting program code is more intuitive. For example :</p> <ul style="list-style-type: none"> • checking for null is now: <code>field.isNull() --> field==null</code> • Converting string to date: <code>field.Clone().str2dat() --> str2date(field)</code> • and so on... |

| Step | Tip | Description |
|---------------|---|---|
| | | <p>If you convert your code as shown above, you may get significant performance benefits.</p> <p> Note: It is no longer possible to modify data in-place using the value methods. This was a design decision to ensure that no data with the wrong type would end up in the output rows of the step. Instead of modifying fields in-place, create new fields using the table at the bottom of the Modified JavaScript transformation.</p> |
| JS | Combine steps | One large JavaScript step runs faster than three consecutive smaller steps. Combining processes in one larger step helps to reduce overhead. |
| JS | Avoid the JavaScript step or write a custom plug in | Remember that while JavaScript is the fastest scripting language for Java, it is still a scripting language. If you do the same amount of work in a native step or plugin, you avoid the overhead of the JS scripting engine. This has been known to result in significant performance gains. It is also the primary reason why the Calculator step was created — to avoid the use of JavaScript for simple calculations. |
| JS | Create a copy of a field | No JavaScript is required for this; a "Select Values" step does the trick. You can specify the same field twice. Once without a rename, once (or more) with a rename. Another trick is to use $B=NVL(A,A)$ in a Calculator step where B is forced to be a copy of A. In version 3.1, an explicit "create copy of field A" function was added to the Calculator. |
| JS | Data conversion | Consider performing conversions between data types (dates, numeric data, and so on) in a "Select Values" step (version 3.0.2 or higher). You can do this in the Metadata tab of the step. |
| JS | Variable creation | If you have variables that can be declared once at the beginning of the transformation, make sure you put them in a separate script and mark that script as a startup script (right click on the script name in the tab). JavaScript object creation is time consuming so if you can avoid creating a new object for every row you are transforming, this will translate to a performance boost for the step. |
| N/A | Launch several copies of a step | <p>There are two important reasons why launching multiple copies of a step may result in better performance:</p> <ol style="list-style-type: none"> 1. The step uses a lot of CPU resources and you have multiple processor cores in your computer. Example: a JavaScript step 2. Network latencies and launching multiple copies of a step can reduce average latency. If you have a low network latency of say 5ms and you need to do a round trip to the database, the maximum performance you get is 200 (x5) rows per second, even if the database is running smoothly. You can try to reduce the round trips with caching, but if not, you can try to run multiple copies. Example: a database lookup or table output |
| N/A | Manage thread priorities | In versions 3.0.2 and higher, this feature that is found in the "Transformation Settings" dialog box under the (Misc tab) improves performance by reducing the locking overhead in certain situations. This feature is enabled by default for new transformations that are created in recent versions, but for older transformations this can be different. |
| Select Value | If possible, don't remove fields in Select Value | Don't remove fields in Select Value unless you must. It's a CPU-intensive task as the engine needs to reconstruct the complete row. It is almost always faster to add fields to a row rather than delete fields from a row. |
| Get Variables | Watch your use of Get Variables | May cause bottlenecks if you use it in a high-volume stream (accepting input). To solve the problem, take the "Get Variables" step out of the transformation (right click, detach) then insert it in with a "Join Rows (cart prod)" step. Make sure to specify the main step from which to read in the "Join Rows" step. Set it to the step that originally provided the "Get Variables" step with data. |
| N/A | Use new text file input | The new "CSV Input" or "Fixed Input" steps provide optimal performance. If you have a fixed width (field/row) input file, you can even read data in parallel. (multiple copies) These new steps have been rewritten using Non-blocking I/O (NIO) features. Typically, the larger the NIO buffer you specify in the step, the better your read performance will be. |
| N/A | When appropriate, use lazy conversion | In instances in which you are reading data from a text file and you write the data back to a text file, use Lazy conversion to speed up the process. The |

| Step | Tip | Description |
|-----------|---|--|
| | | principle behind lazy conversion that it delays data conversion in hopes that it isn't necessary (reading from a file and writing it back comes to mind). Beyond helping with data conversion, lazy conversion also helps to keep the data in "binary" storage form. This, in turn, helps the internal Kettle engine to perform faster data serialization (sort, clustering, and so on). The Lazy Conversion option is available in the "CSV Input" and "Fixed input" text file reading steps. |
| Join Rows | Use Join Rows | You need to specify the main step from which to read. This prevents the step from performing any unnecessary spooling to disk. If you are joining with a set of data that can fit into memory, make sure that the cache size (in rows of data) is large enough. This prevents (slow) spooling to disk. |
| N/A | Review the big picture: database, commit size, row set size and other factors | Consider how the whole environment influences performance. There can be limiting factors in the transformation itself and limiting factors that result from other applications and PDI. Performance depends on your database, your tables, indexes, the JDBC driver, your hardware, speed of the LAN connection to the database, the row size of data and your transformation itself. Test performance using different commit sizes and changing the number of rows in row sets in your transformation settings. Change buffer sizes in your JDBC drivers or database. |
| N/A | Step Performance Monitoring | Step Performance Monitoring is an important tool that allows you identify the slowest step in your transformation. |

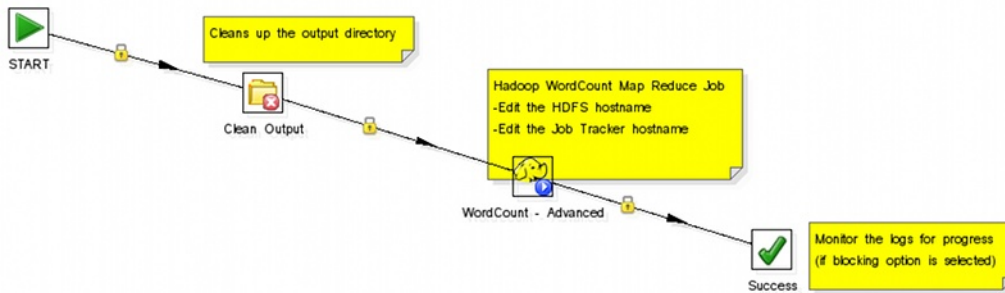
Working With Hadoop

This section contains guidance and instructions on the Hadoop-related functionality in Pentaho Data Integration. Hadoop job and transformation steps are covered in the standard PDI job and step reference sections in this guide.

 **Note:** Some Hadoop functionality is exclusive to the Pentaho BI Suite For Hadoop. If you require advanced Hadoop functionality, contact your Pentaho sales representative.

Hadoop Job Process Flow

There are two paradigms for jobs in PDI: native PDI jobs, which are processes that typically include running transformations or other jobs; and Hadoop jobs, which are executed on the Hadoop node containing the data you are working with. PDI has the ability to design and execute Hadoop jobs in a similar manner to native PDI jobs. The relevant step is called **Hadoop Job Executor**:



This step requires a custom mapper/reducer Java class:

Hadoop Job Executor

Name: WordCount - Simple

Hadoop Job Name: PDI Hadoop - WordCount - Simple

Jar: ./samples/jobs/hadoop/pentaho-mapreduce-sample.jar Browse...

Configuration

Simple Advanced

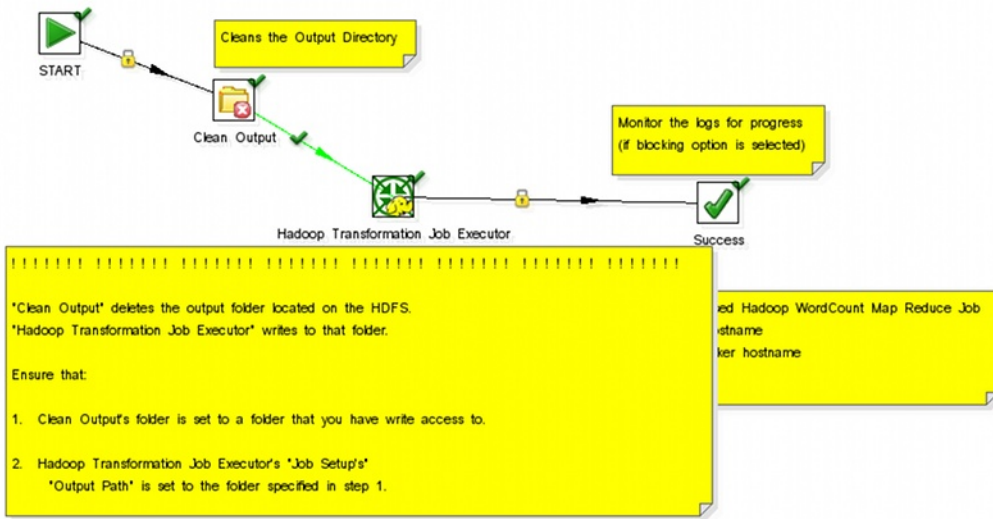
Command line arguments: /junit/wordcount/input /junit/wordcount/pdioutput

OK Cancel

If you are using the Amazon Elastic MapReduce (EMR) service, you can use a similar Hadoop job step called **Amazon EMR Job Executor**. This differs from the standard Hadoop Job Executor in that it contains connection information for Amazon S3 and configuration options for EMR:

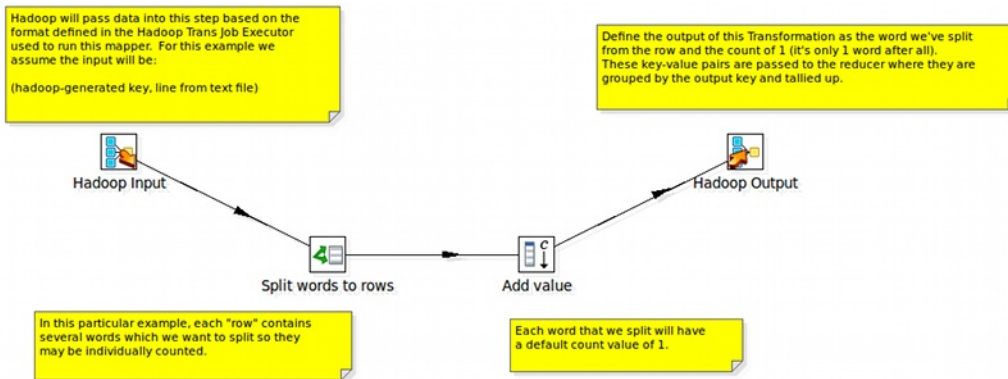
You can also execute a PDI job that includes Hadoop-oriented transformations through the **Hadoop Transformation Job Executor**. In addition to ordinary transformation work, you can also use this step to design mapper/reducer functions within PDI, removing the need to provide a Java class. To do this, you must create transformations that act as a mapper and a reducer, then reference them properly in the step configuration:

The workflow for the transformation job executor looks something like this:



Hadoop Transformation Process Flow

Pentaho Data Integration enables you to pull data from a Hadoop cluster, transform it in any of the usual ways, and pass it back to the cluster. You can also use specially-designed transformations as Hadoop mappers and reducers, which completely removes the need to create a Java class for these purposes. However, you must follow a specific workflow in order to properly communicate with Hadoop, as shown in this sample transformation:



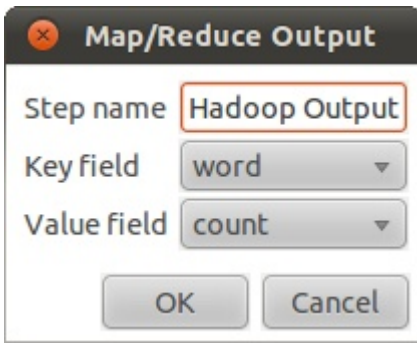
Hadoop will only communicate in terms of key/value pairs. Therefore, PDI must use a **Map/Reduce Input** step that defines the data type and name of the key and value:

Map/Reduce Input

Step name:

| | Type | Length | Precision |
|-------------|--------|--------|-----------|
| Key field | String | 0 | 0 |
| Value field | String | 0 | 0 |

...and a **Map/Reduce Output** step that passes the output back to Hadoop.



What happens in the middle is entirely up to the user.

Hadoop to PDI Data Type Conversion

The Hadoop Job Executor and Hadoop Transformation Job Executor steps have an advanced configuration mode that allows you to specify data types for the job's input and output. PDI is unable to detect foreign data types on its own; therefore you must specify the input and output data types in the **Job Setup** tab. The table below explains the relationship between Apache Hadoop data types and their PDI equivalents.

| PDI (Kettle) Data Type | Apache Hadoop Data Type |
|-----------------------------------|-----------------------------------|
| java.lang.Integer | org.apache.hadoop.io.IntWritable |
| java.lang.Long | org.apache.hadoop.io.IntWritable |
| java.lang.Long | org.apache.hadoop.io.LongWritable |
| org.apache.hadoop.io.IntWritable | java.lang.Long |
| java.lang.String | org.apache.hadoop.io.Text |
| java.lang.String | org.apache.hadoop.io.IntWritable |
| org.apache.hadoop.io.LongWritable | org.apache.hadoop.io.Text |
| org.apache.hadoop.io.LongWritable | java.lang.Long |

Interacting With Web Services

PDI jobs and transformations can interact with a variety of Web services through specialized steps. How you use these steps, and which ones you use, is largely determined by your definition of "Web services." The most commonly used Web services steps are:

- [Web Service Lookup](#)
- [Modified Java Script Value](#)
- [RSS Input](#)
- [HTTP Post](#)

The Web Service Lookup Step is useful for selecting and setting input and output parameters via WSDL, but only if you do not need to modify the SOAP request. You can see this step in action in the **Web Services - NOAA Latitude and Longitude.ktr** sample transformation included with PDI in the `/data-integration/samples/transformations/` directory.

There are times when the SOAP message generated by the Web Services Lookup step is insufficient. Many Web services require the security credentials be placed in the SOAP request headers. There may also be a need to parse the response XML to get more information than the response values provide (such as namespaces). In cases like these, you can use the Modified Java Script Value step to create whatever SOAP envelope you need. You would then hop to an HTTP Post step to accept the SOAP request through the input stream and post it to the Web service, then hop to another Modified Java Script Value to parse the response. The **General - Annotated SOAP Web Service call.ktr** sample in the `/data-integration/samples/transformations/` directory shows this theory in practice.

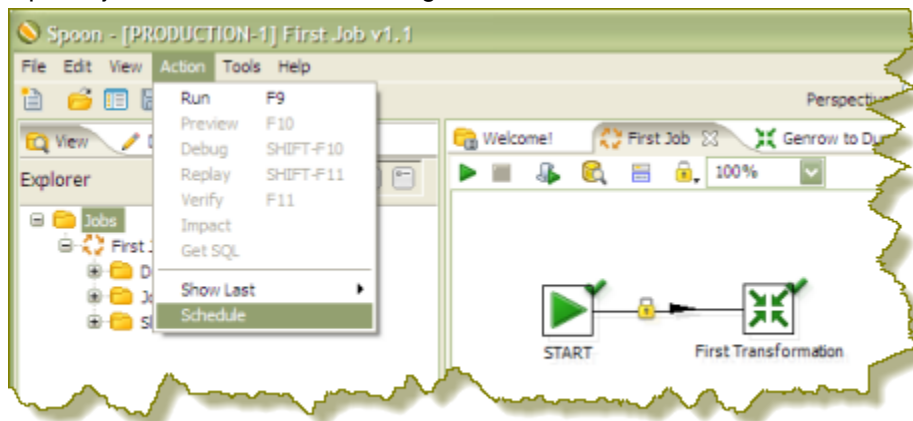
Scheduling and Scripting PDI Content

Once you're finished designing your PDI jobs and transformations, you can arrange to run them at certain time intervals through the DI Server, or through your own scheduling mechanism (such as **cron** on Linux, and the **Task Scheduler** or the **at** command on Windows). The methods of operation for scheduling and scripting are different; scheduling through the DI Server is done through the Spoon graphical interface, whereas scripting using your own scheduler or executor is done by calling the **pan** or **kitchen** commands. This section explains all of the details for scripting and scheduling PDI content.

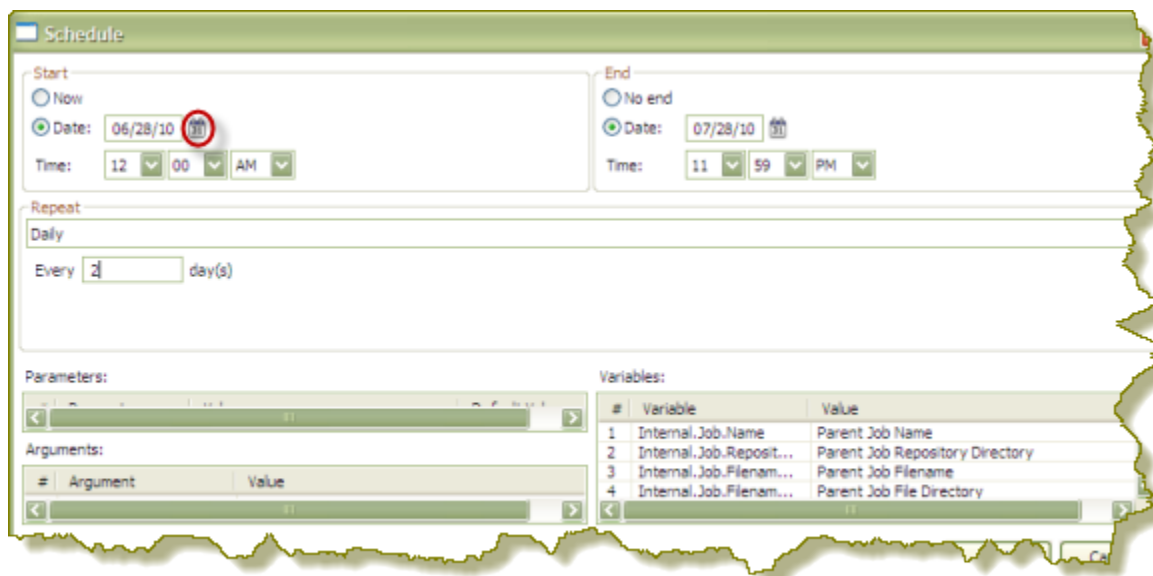
Scheduling Transformations and Jobs From Spoon

You can schedule jobs and transformations to execute automatically on a recurring basis by following the directions below.

1. Open a job or transformation, then go to the **Action** menu and select **Schedule**.



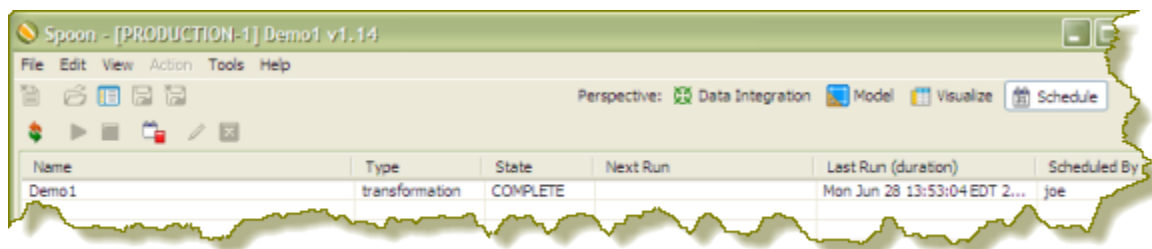
2. In the **Schedule a Transformation** dialog box, enter the date and time that you want the schedule to begin in the **Start** area, or click the calendar icon (circled in red) to display the calendar. To run the transformation immediately, enable the **Now** radio button.



3. Set up the **End** date and time. If applicable, enable the **No end** radio button or click on the calendar and input the date and time to end the transformation.
4. If applicable, set up a recurrence under **Repeat**.

End date and time are disabled unless you select a recurrence. From the list of schedule options select the choice that is most appropriate: **Run Once**, **Seconds**, **Minutes**, **Hourly**, **Daily**, **Weekly**, **Monthly**, **Yearly**.

5. Make sure you set parameters, arguments and variables, if available. Click **OK**.
6. In the Spoon button bar, click the **Schedule** perspective.



From the Schedule perspective, you can refresh, start, pause, stop and delete a transformation or job using the buttons on the upper left corner of the page.



Command-Line Scripting Through Pan and Kitchen

You can use PDI's command line tools to execute PDI content from outside of Spoon. Typically you would use these tools in the context of creating a script or a cron job to run the job or transformation based on some condition outside of the realm of Pentaho software.

Pan is the PDI command line tool for executing transformations.

Kitchen is the PDI command line tool for executing jobs.

Both of these programs are explained in detail below.

Pan Options and Syntax

Pan runs transformations, either from a PDI repository (database or enterprise), or from a local file. The syntax for the batch file and shell script are shown below. All Pan options are the same for both.

```
pan.sh - option = value arg1 arg2
```

```
pan.bat /option:value arg1 arg2
```

| Switch | Purpose |
|-----------|---|
| rep | Enterprise or database repository name, if you are using one |
| user | Repository username |
| pass | Repository password |
| trans | The name of the transformation (as it appears in the repository) to launch |
| dir | The repository directory that contains the transformation, including the leading slash |
| file | If you are calling a local KTR file, this is the filename, including the path if it is not in the local directory |
| level | The logging level (Basic, Detailed, Debug, Rowlevel, Error, Nothing) |
| logfile | A local filename to write log output to |
| listdir | Lists the directories in the specified repository |
| listtrans | Lists the transformations in the specified repository directory |
| listrep | Lists the available repositories |
| exprep | Exports all repository objects to one XML file |

| Switch | Purpose |
|---------------|--|
| norep | Prevents Pan from logging into a repository. If you have set the KETTLE_REPOSITORY, KETTLE_USER, and KETTLE_PASSWORD environment variables, then this option will enable you to prevent Pan from logging into the specified repository, assuming you would like to execute a local KTR file instead. |
| safemode | Runs in safe mode, which enables extra checking |
| version | Shows the version, revision, and build date |
| param | Set a named parameter in a name=value format. For example: -param:FOO=bar |
| listparam | List information about the defined named parameters in the specified transformation. |
| maxloglines | The maximum number of log lines that are kept internally by PDI. Set to 0 to keep all rows (default) |
| maxlogtimeout | The maximum age (in minutes) of a log line while being kept internally by PDI. Set to 0 to keep all rows indefinitely (default) |

```
sh pan.sh -rep=initech_pdi_repo -user=pgibbons -pass=lumburghsux -
trans=TPS_reports_2011
```

```
pan.bat /rep:initech_pdi_repo /user:pgibbons /pass:lumburghsux /
trans:TPS_reports_2011
```

Pan Status Codes

When you run Pan, there are seven possible return codes that indicate the result of the operation. All of them are defined below.

| Status code | Definition |
|-------------|---|
| 0 | The transformation ran without a problem. |
| 1 | Errors occurred during processing |
| 2 | An unexpected error occurred during loading / running of the transformation |
| 3 | Unable to prepare and initialize this transformation |
| 7 | The transformation couldn't be loaded from XML or the Repository |
| 8 | Error loading steps or plugins (error in loading one of the plugins mostly) |
| 9 | Command line usage printing |

Kitchen Options and Syntax

Kitchen runs jobs, either from a PDI repository (database or enterprise), or from a local file. The syntax for the batch file and shell script are shown below. All Kitchen options are the same for both.

`kitchen.sh` - option = value arg1 arg2

`kitchen.bat` / option : value arg1 arg2

| Switch | Purpose |
|---------------|--|
| rep | Enterprise or database repository name, if you are using one |
| user | Repository username |
| pass | Repository password |
| job | The name of the job (as it appears in the repository) to launch |
| dir | The repository directory that contains the job, including the leading slash |
| file | If you are calling a local KJB file, this is the filename, including the path if it is not in the local directory |
| level | The logging level (Basic, Detailed, Debug, Rowlevel, Error, Nothing) |
| logfile | A local filename to write log output to |
| listdir | Lists the directories in the specified repository |
| listjob | Lists the jobs in the specified repository directory |
| listrep | Lists the available repositories |
| export | Exports all linked resources of the specified job. The argument is the name of a ZIP file. |
| norep | Prevents Kitchen from logging into a repository. If you have set the KETTLE_REPOSITORY, KETTLE_USER, and KETTLE_PASSWORD environment variables, then this option will enable you to prevent Kitchen from logging into the specified repository, assuming you would like to execute a local KTR file instead. |
| version | Shows the version, revision, and build date |
| param | Set a named parameter in a name=value format. For example: - param:FOO=bar |
| listparam | List information about the defined named parameters in the specified job. |
| maxloglines | The maximum number of log lines that are kept internally by PDI. Set to 0 to keep all rows (default) |
| maxlogtimeout | The maximum age (in minutes) of a log line while being kept internally by PDI. Set to 0 to keep all rows indefinitely (default) |

```
sh kitchen.sh -rep=initech_pdi_repo -user=pgibbons -pass=lumburghsux -  
job=TPS_reports_2011
```

```
kitchen.bat /rep:initech_pdi_repo /user:pgibbons /pass:lumburghsux /  
job:TPS_reports_2011
```

Kitchen Status Codes

When you run Kitchen, there are seven possible return codes that indicate the result of the operation. All of them are defined below.

| Status code | Definition |
|-------------|---|
| 0 | The job ran without a problem. |
| 1 | Errors occurred during processing |
| 2 | An unexpected error occurred during loading or running of the job |
| 7 | The job couldn't be loaded from XML or the Repository |
| 8 | Error loading steps or plugins (error in loading one of the plugins mostly) |
| 9 | Command line usage printing |

Transformation Step Reference

There are over 140 transformation steps associated with Pentaho Data Integration. The listing below is a subset of the most commonly-used steps. In later versions of this document, more steps will be added to the list. Currently, the bulk of step-related documentation is available in the [Pentaho Wiki](#); however, Wiki documents are maintained by the open source community and are therefore not always as comprehensive or as accurate.

Hadoop

| Name | Description |
|------------------------------------|---|
| Hadoop File Input | Retrieves data stored on a Hadoop node. |
| Hadoop File Output | Stores data in a file on a Hadoop node. |

Input

| Name | Description |
|----------------------------------|--|
| CSV Input | Reads data from a delimited file |
| Excel File Input | Reads data from a Microsoft Excel or OpenOffice.org Calc file |
| Fixed File Input | Reads data from a fixed width file |
| Generate Rows | Outputs a specified number of rows |
| Google Analytics | Accesses your Google analytics data to generate reports or to populate your BI data warehouse |
| Google Docs | Reads data from one or more Google Docs spreadsheets so you can populate Pentaho reports, dashboards, and charts |
| JMS Consumer | Allows Pentaho Data Integration to receive messages from any JMS server |
| Table Input | Reads information from a database, using a connection and SQL. Basic SQL statements are generated automatically. |
| Text File Input | Reads data from a variety of different text-file types |

Output

| Name | Description |
|------------------------------------|---|
| Excel File Output | Exports data to a Microsoft Excel file |
| Hadoop File Output | Exports data to text files stored on a Hadoop node |
| JMS Producer | Allows Pentaho Data Integration to send messages to any JMS server. |
| Table Output | Loads data into a database table |
| Text File Output | Exports data to text file format |

Transform

| Name | Description |
|-------------------------------|--|
| Select Values | Selects, renames, changes data types and configures the length and precision of the fields in the stream |

Flow

| Name | Description |
|------------------------------------|---|
| Dummy (do nothing) | Placeholder for testing purposes |
| Filter Rows | Filter rows based on conditions and comparisons |

Lookup

| Name | Description |
|---------------------------------|-------------------------------------|
| Database Lookup | Looks up values in a database table |

| Name | Description |
|-------------------------------------|---|
| Stream Lookup | Looks up data using information coming from other steps in the transformation |
| Web Services Lookup | Performs a Web Services lookup using the Web Services Description Language (WSDL) |

Joins

| Name | Description |
|----------------------------|--|
| Join Rows | Produces combinations (Cartesian product) of all rows in the input streams |
| Merge Rows | Compares two streams of rows |

Data Warehouse

| Name | Description |
|---|---|
| Combination Lookup/Update | Stores information in a junk-dimension table |
| Dimension Lookup/Update | Implements the Kimball slowly changing dimension for both types: Type1 (update) and Type 2 (insert) |

Statistics

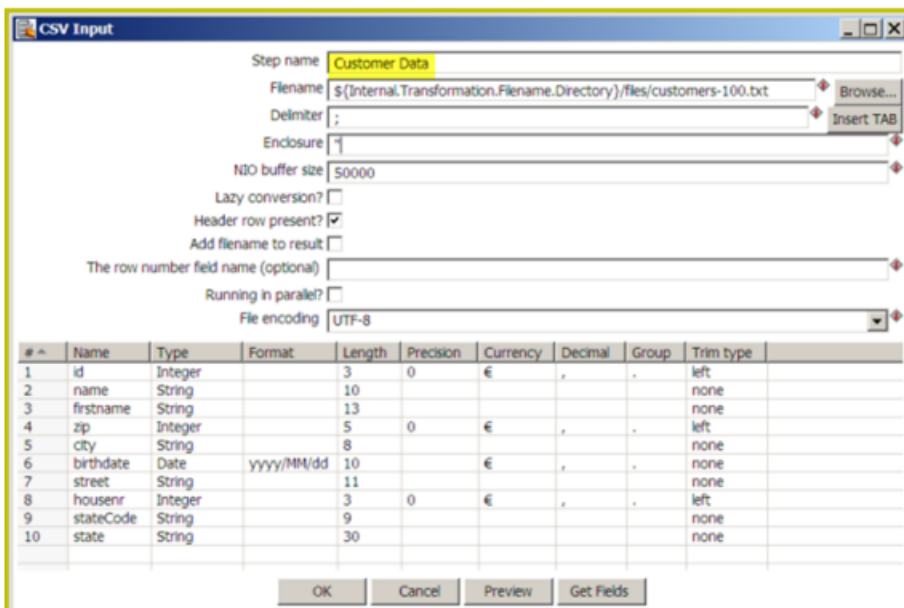
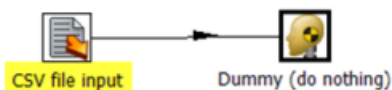
| Name | Description |
|--------------------------|--|
| Group By | Calculates values over a defined group of fields |

Scripting

| Name | Description |
|---|--|
| Modified JavaScript Value | Performs complex calculations using JavaScript |


CSV File Input

The CSV File Input step reads a *delimited* file format. The CSV label for this step is a misnomer because you can define whatever separator you want to use, such as pipes, tabs, and semicolons; you are not constrained to using commas. Internal processing allows this step to process data quickly. Options for this step are a subset of the Text File Input step. An example of a simple CSV Input transformation can be found under `... \samples \transformations \CSV Input - Reading customer data.ktr`.



CSV File Input Options

The table below describes the options available for the CSV Input step:

| Option | Description |
|---|--|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| File Name | Specify the name of the CSV file from which to read or select the field name that will contain the file name(s) from which to read. If your CSV Input step receives data from a previous step, this option is enabled as well as the option to include the file name in the output. |
| Delimiter | Specify the file delimiter or separator used in the target file. This includes pipes, tabs, semicolons and so on. In the sample image, the delimiter is a semicolon. |
| Enclosure | Specify the enclosure character used in the target file. It's possible that your strings contain semicolons or commas as delimiters, so the enclosures specify that a textual string inside an enclosure, such as a "quotation mark" is not to be parsed until the "end" enclosure. In the sample image, the enclosure is a quotation mark. |
| NIO buffer size | The size of the read buffer. It represents the number of bytes that is read at one time from disk. |
| Lazy conversion | Lazy conversion delays conversion of data as long as possible. In some instances, data conversion is prevented altogether. This can result in significant performance improvements when possible. The typical example that comes to mind is reading from a text file and writing back to a text file. |
| Header row present? | Enable this option if the target file contains a header row containing column names. Header rows are skipped. |
| Add file name to result | Adds the CSV filename(s) read to the result of this transformation. A unique list is being kept in memory that can be used in the next job entry in a job, for example in another transformation. |
| The row number field name (optional) | The name of the Integer field that will contain the row number in the output of this step. |
| Running in parallel? | <p>Enable if you will have multiple instances of this step running (step copies) and if you want each instance to read a separate part of the CSV file(s).</p> <p>When reading multiple files, the total size of all files is taken into consideration to split the workload. In that specific case, make sure that ALL step copies receive all files that need to be read, otherwise, the parallel algorithm will not work correctly (for obvious reasons).</p> <p> Note: For technical reasons, parallel reading of CSV files is supported only for files that do not include fields with line breaks or carriage returns.</p> |
| File Encoding | Specify the encoding of the file being read. |
| Fields Table | This table contains an ordered list of fields to be read from the target file. |
| Preview | Click to preview the data coming from the target file. |
| Get Fields | Click to return a list of fields from the target file based on the current settings (for example, Delimiter, Enclosure, and so on.). All fields identified will be added to the Fields Table. |

Excel File Input

This step imports data from a Microsoft Excel (2003 or 2007) or OpenOffice.org Calc spreadsheet file.



Note: The **Files**, **Sheets**, and **Fields** tabs are required for proper step configuration.

Files Tab

The Files tab defines basic file properties for this step's output.

| Option | Description |
|-------------------------------------|--|
| Step name | The name of this step in the transformation workspace. |
| File or directory | The name of the spreadsheet file or directory of files that you are reading from. |
| Regular Expression | Includes all files (in a given location) that meet the criteria specified by this regular expression. |
| Exclude Regular Expression | Excludes all files (in a given location) that meet the criteria specified by this regular expression. |
| Selected files | A list of files that will be used in this step, according to the criteria specified in the previous fields. |
| Accept filenames from previous step | If checked, will retrieve a list of filenames from the previous step in this transformation. You must also specify which step you are importing from, and the input field in that step from which you will retrieve the filename data. If you choose this option, the Show filename(s) option will show a preview of the list of filenames. |

Sheets Tab

The Sheets tab specifies which worksheets you want to use in the specified files. A spreadsheet document can contain several worksheets.

| Option | Description |
|------------------------|---|
| List of sheets to read | A list of worksheets that you want to use. If this remains empty, all worksheets in all specified files will be selected. Rows and columns are numbered, starting with 0. |
| Get sheetname(s) | This button will retrieve a list of worksheets from all of the specified files and give you the option to select some or all of them for this step. |

Content tab

The content tab contains options for describing the file's content.

| Option | Description |
|--------|---|
| Header | Enable this option if there is a header row to skip in the selected worksheets. |

| Option | Description |
|---------------------------|---|
| No empty rows | If checked, removes empty rows from the output. |
| Stop on empty row | If checked, stops reading from the current worksheet when an empty row is read. |
| Limit | Sets a static number of rows to read. If set to 0, there is no set limit. |
| Encoding | Specifies 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 searches your system for available encodings. |
| Spreadsheet type (engine) | Specifies which spreadsheet format to expect from the file, regardless of its extension. |
| Add filenames to result | If checked, passes the input filenames to the output. |

Error Handling tab

This tab sets options for recording and reporting various error conditions.

| Option | Description |
|--------------------------------------|--|
| Strict types? | If checked, PDI will report data type errors in the input. |
| Ignore errors? | If checked, no errors will be reported during input parsing. |
| Skip error lines? | If checked, PDI will skip lines that contain errors. These lines can be dumped to a separate file by specifying a path in the Failing line numbers files directory field below. If this is not checked, lines with errors will appear as NULL values in the output. |
| Warning files directory | Directory in which to create files that contain warning messages regarding input values for each spreadsheet file read. These files will have the extension you specify here. |
| Error files directory | Directory in which to create files that contain error messages regarding input values for each spreadsheet file read. These files will have the extension you specify here. |
| Failing line numbers files directory | Directory in which to create files that contain the lines that failed error checks during input validation. These files will have the extension you specify here. |

Fields tab

The Fields tab defines properties for the exported fields.

| Option | Description |
|--------|--|
| Name | The name of the field. |
| Type | The field's data type; String, Date or Number. |

| Option | Description |
|-----------|--|
| Length | The length option depends on the field type. Number : total number of significant figures in a number; String : total length of a string; Date : determines how much of the date string is printed or recorded. |
| Precision | The precision option depends on the field type, but only Number is supported; it returns the number of floating point digits. |
| Trim type | Truncates the field (left, right, both) before processing. Useful for fields that have no static length. |
| Repeat | If set to Y , will repeat this value if the next field is empty. |
| Format | The format mask (number type). |
| Currency | Symbol used to represent currencies. |
| Decimal | A decimal point; this is either a dot or a comma. |
| Grouping | A method of separating units of thousands in numbers of four digits or larger. This is either a dot or a comma. |

Additional output fields tab

This tab retrieves custom metadata fields to add to the step's output. The purpose of each field is defined in its name, but you can use these fields for whatever you want. Each item defines an output field that will contain the following information:

Excel File Output

This step exports data to a Microsoft Excel 2003 spreadsheet file.

File Tab

The File tab defines basic file properties for this step's output.

| Option | Description |
|-----------------------------|--|
| Step name | The name of this step in the transformation workspace. |
| Filename | The name of the spreadsheet file you are reading from. |
| Do not create file at start | If checked, does not create the file until the end of the step. |
| Extension | The three-letter file extension to append to the file name. |
| Include stepnr in filename | If you run the step in multiple copies (launching several copies of a step), the copy number is included in the file name, before the extension. (_0). |
| Include date in file name | Includes the system date in the filename (_20101231). |
| Include time in file name | Includes the system time (24-hour format) in the filename (_235959). |
| Specify Date time format | If checked, the filename will include a date and time stamp that follows the selection you choose from the |

| Option | Description |
|-------------------------|--|
| | drop-down box. Selecting this option disables the previous two options. |
| Show file name(s) | Displays a list of the files that will be generated. This is a simulation and depends on the number of rows that will go into each file. |
| Add filenames to result | Uses the Filename field in constructing the result filename. If un-checked, the Filename field is ignored. |

Content tab

The content tab contains options for describing the file's content.

| Option | Description |
|--------------------------|---|
| Append | When checked, appends lines to the end of the specified file. If the file does not exist, a new one will be created. |
| Header | Enable this option if you want a header to appear before the spreadsheet grid data. |
| Footer | Enable this option if you want a footer to appear after the spreadsheet grid data. |
| Encoding | Specifies 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 searches your system for available encodings and populates this list accordingly. |
| Split every ... rows | After this many rows, start a new spreadsheet file to continue data output. |
| Sheet name | Specifies the name of the worksheet within the spreadsheet file. |
| Protect sheet? | If checked, enables password protection on the worksheet. You must also specify a password in the Password field. |
| Auto size columns | If checked, automatically sizes the worksheet columns to the largest value. |
| Retain NULL values | If checked, NULL values are preserved in the output. If un-checked, NULLs are replaced with empty strings. |
| Use Template | If checked, PDI will use the specified Excel template to create the output file. The template must be specified in the Excel template field. |
| Append to Excel Template | Appends output to the specified Excel template. |

Fields tab

The Fields tab defines properties for the exported fields. The **Get Fields** button will automatically retrieve a list of fields from the inputstream and populate the list. The **Minimal width** button removes any padding from the output.

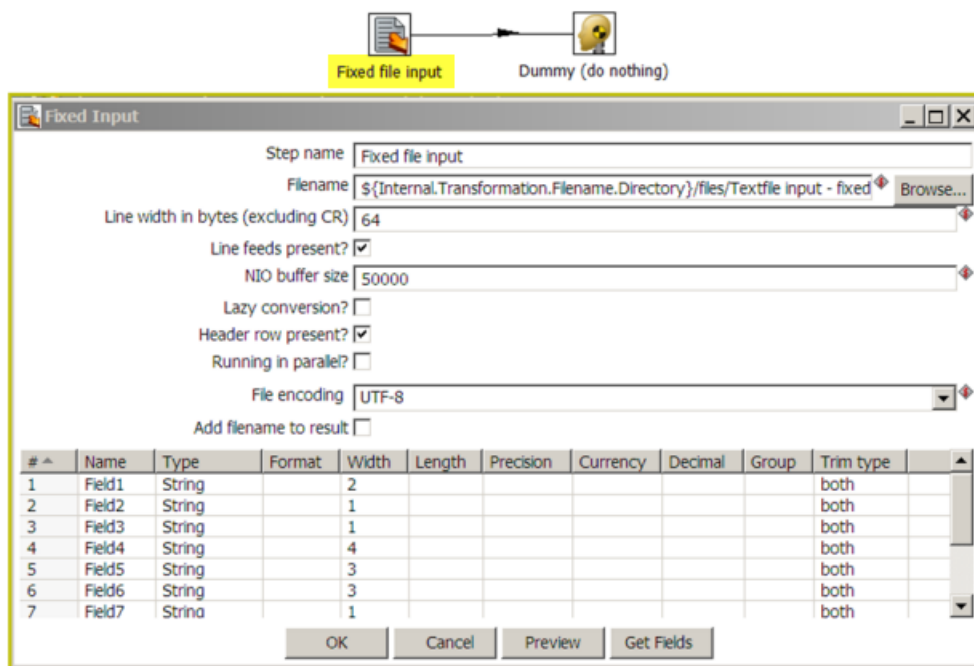
| Option | Description |
|--------|------------------------|
| Name | The name of the field. |

| Option | Description |
|--------|--|
| Type | The field's data type; String, Date or Number. |
| Format | The format mask (number type). |

Fixed File Input Step

This step is used to read data from a fixed-width text file, exclusively. In fixed-width files, the format is specified by column widths, padding, and alignment. Column widths are measured in units of characters. For example, the data in the file contains a first column that has exactly 12 characters, and the second column has exactly 10, the third has exactly 7, and so on. Each row contains one record of information; each record can contain multiple pieces of data (fields), each data field (column) has a specific number of characters. When the data does not use all the characters allotted to it, the data is padded with spaces (or other character). In addition, each data element may be left or right justified, which means that characters can be padded on either side.

A sample Fixed File Input transformation is located at `... \samples \transformations \Fixed Input - fixed length reading .ktr`



The table below describes the options available for the Fixed File Input step:

Fixed File Options

| Option | Description |
|--------------------------------|---|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| File name | Specify the CSV file from which to read. |
| Line feeds present? | Enable if the target file contains line feed characters; line width in bytes (excluding carriage returns) — defines the width of each line in the input file |
| NIO buffer size | The size of the read buffer — represents the number of bytes that is read at one time from disk |
| Lazy conversion | The lazy conversion algorithm will try to avoid unnecessary data type conversions and can result in a significant performance improvements if this is possible. The typical example that comes to mind is reading from a text file and writing back to a text file. |
| Header row present? | Enable if the target file contains a header row containing column names. |
| Running in parallel? | Enable if you will have multiple instances of this step running (step copies) and if you want each instance to read a separate part of the file. |
| File Encoding | Specify the encoding of the file being read. |
| Add file name to result | Adds the file name(s) read to the result of this transformation. A unique list is kept in memory so that it can be used in the next job entry in a job, for example in another transformation. |

| Option | Description |
|---------------------|---|
| Fields Table | Contains an ordered list of fields to be read from the target file. |
| Preview | Click to preview the data coming from the target file. |
| Get Fields | Click to return a list of fields from the target file based on the current settings; for example, Delimiter, Enclosure, and so on. All fields identified will be added to the Fields Table. |

Generate Rows


Generate rows outputs a specified number of rows. By default, the rows are empty; however they can contain a number of static fields. This step is used primarily for testing purposes. It may be useful for generating a fixed number of rows, for example, you want exactly 12 rows for 12 months. Sometimes you may use Generate Rows to generate one row that is an initiating point for your transformation. For example, you might generate one row that contains two or three field values that you might use to parameterize your SQL and then generate the real rows.

Generate Rows Options

| Option | Description |
|------------------|---|
| Step Name | Optionally, you can change the name of this step to fit your needs |
| Limit | Specifies the number of rows to output |
| Fields | This table is where you configure the structure and values of the rows you are generating (optional). This may be used to generate constants. |

Google Analytics Input Step

The Google Analytics step allow you to access your Google analytics data to generate reports or populate your BI data warehouse.

 **Note:** To make querying easier, a link provides you with quick access to the Google Analytics API documentation.

Authorization

| Option | Description |
|-----------------|------------------------------------|
| Username | Google Analytics account user name |
| Password | Google Analytics account password |

Query

| Option | Description |
|------------------------|---|
| Domain Table ID | Specifies the domain associated with Google Analytics that must be queried. Click Lookup to display the list of available domains. |
| Start Date | Specifies the start date associated with the query - date must be entered in the following format: year, month, and date (for example, 2010-03-01) |
| End Date | Specifies the end date associated with the query - date must be entered in the following format: year, month, and date (for example, 2010-03-31) |
| Dimensions | Specifies the dimension fields for which you want to query - the Google Analytics API documentation provides you with a list of valid inputs and metrics that can be combined |
| Metrics | Specifies the metrics fields you want returned |
| Filters | Specifies the filter (described in the Google Analytics API documentation) for example, 'ga:country==Algeria' |
| Sort | Specifies a field on which to sort, for example, 'ga:city' |

Fields

Click **Get Fields** to retrieve a list of possible fields based on the query you defined on the Query tab.

Click **Preview Rows** to preview data based on the defined query.

Google Docs Input

The Google Docs Input step provides you with the ability to read data from one or more Google Docs spreadsheets. The following sections describe each of the available features for configuring the Google Docs Input step. If necessary, you refer to the Google [Dimensions and Metrics Reference](#).

Files

The Files tab is where you define the location of the Google Docs files that you want read. The table below contains options associated with the Files tab:

| Option | Description |
|------------------------------|---|
| Step Name | Optionally, you can change the name of this step to fit your needs. |
| Username | Google Docs account user name |
| Password | Google Docs account password |
| Google Docs Object ID | Key to the Google document from which you want to read data - Note: The key is included in the URL associated with the document; your entry must be in the following format spreadsheet%pBb5yoxtYzKEyXDB9eqsNVG. Click Lookup to display the list of available keys. |

Sheets

The options in the Sheets tab allow you to specify the names of the sheets in the Google Docs workbook to read. For each of the sheet names, you can specify the row and column to start at. The row and column numbers are zero (0) based; start number is 0.

Content

The content tab allows you to configure the following properties:

| Option | Description |
|----------------------------|---|
| Header | Enable if the sheets specified contain a header row to skip |
| No empty rows | Enable if you don't want empty rows in the output of this step |
| Stop on empty row | Makes the step stop reading the current sheet of a file when a empty line is encountered |
| Filename field | Specifies a field name to include the file name in the output of this step |
| Sheetname field | Specifies a field name to include the sheet name in the output of this step |
| Sheet row nr field | Specifies 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 Google Docs sheet |
| Row nrwritten field | Specifies 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 indefinitely |
| Limit | Limits the number of rows to this number (zero (0) means all rows). |
| Encoding | Specifies the character encoding (such as UTF-8, ASCII) |

Error Handling

The Error handling tab allows you to configure the following properties:

| Option | Description |
|---|--|
| Strict types? | Certain columns in the Google Docs input step can be flagged as numbers, strings, dates, and so on. Once flagged, if a column does not contain the right data type; for example, the column was flagged as numeric but contains a string input, an error occurs. |
| Ignore errors? | Enable if you want to ignore errors during parsing |
| Skip error lines? | Enable if you want to skip the lines that contain errors. Note: 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). |
| Warnings file directory | When warnings are generated, they are placed in this directory. The name of that file is <warning_dir>/filename.<date_time>.<warning extension> |
| Error files directory | When errors occur, they are placed in this directory. The name of that file is <errorfile_dir>/filename.<date_time>.<errorfile_extension> |
| Failing line numbers files directory | When a parsing error occurs on a line, the line number is placed in this directory. The name of that file is <errorline_dir> / filename.<date_time>.<errorline extension> |

Fields

The fields tab is for specifying the fields that must be read from the Google Docs files. Use **Get fields** from header row to fill in the available fields if the sheets have a header row automatically. The Type column performs type conversions for a given field. For example, if you want to read a date and you have a String value in the Google Docs file, specify the conversion mask.


 **Note:** In the case of Number to Date conversion (for example, 20101028--> October 28th, 2010) specify the conversion mask yyyyMMdd because there will be an implicit Number to String conversion taking place before doing the String to Date conversion.

Table Input

This step is used to read information from a database, using a connection and SQL. Basic SQL statements can be generated automatically by clicking **Get SQL select statement**.

Table Input Options


| Option | Description |
|-------------------------------------|---|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Connection | The database connection from which to read data (see Managing Connections) |
| SQL | The SQL statement used to read information from the database connection. You can also click Get SQL select statement... to browse tables and automatically generate a basic select statement. |
| Enable lazy conversion | When enables, lazy conversion avoids unnecessary data type conversions and can result in a significant performance improvements. |
| Replace variables in script? | Enable to replace variables in the script; this feature was provided to allow you to test with or without performing variable substitutions. |
| Insert data from step | Specify the input step name where Pentaho? an expect information to come from. This information can then be inserted into the SQL statement. The locators where |

| Option | Description |
|------------------------------|---|
| | Pentaho? inserts information is indicated by ? (question marks). |
| Execute for each row? | Enable to perform the data insert for each individual row. |
| Limit size | Sets the number of lines that is read from the database; zero (0) means read all lines. |

Example... Below is a sample SQL statement::

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

This statement requires two dates that are read on the Insert data from the step.

 **Note:** The dates can be provided using the **Get System Info** step. For example, if you want to read all customers that have had their data changed yesterday, you may get the range for yesterday and read the customer data

Preview... Preview allows you to preview the step. This is accomplished by preview of a new transformation with two steps: this one and a Dummy step. To see a detailed log of the execution, click **Logs** in the Preview window.

For additional information, see [Table Input](#)

Text File Input

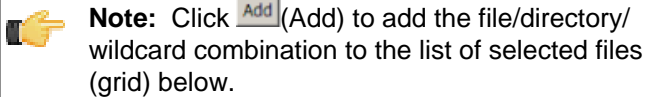
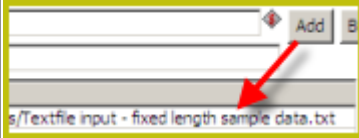
The **Text File Input** step is used to read data from a variety of different text-file types. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

The Text File Input step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept file names from a previous step making file name handling more even more generic.

The following sections describe the available options for configuring the Text file input step. You can find an example of a simple Text File Input transformation at `...\samples\transformations\Text File Output - Number formatting.ktr`.




File Tab Options

| Option | Description |
|--|---|
| Step Name | Optionally, you can change the name of this step to fit your needs. |
| File or Directory | Specifies the location and/or name of the input text file (s) to be read. .   |
| Regular expression | Specify the regular expression you want to use to select the files in the directory specified in the previous option. For example, you want to process all files that have a .txt extension. (See below) |
| Selected Files | This table contains a list of selected files (or wild card selections) along with a property specifying if file is required or not. If a file is required and it isn't found, an error is generated. Otherwise, the file name is skipped. |
| Show filenames(s)... | Displays a list of all files that will be loaded based on the current selected file definitions. |
| Show file content | Displays the raw content of the selected file. |
| Show content from first data line | Displays the content from the first data line only for the selected file. |

 **Note:**

Selecting file using Regular Expressions The Text File Input step can search for files by wildcard in the form of a regular expression. Regular expressions are more sophisticated than using '*' and '?' wildcards. Below are a few examples of regular expressions:

| File Name | Regular Expression | Files selected |
|-----------|----------------------|---|
| /dirA/ | .userdata.\.txt | Find all files in /dirA/ with names containing userdata and ending with .txt |
| /dirB/ | AAA.* | Find all files in /dirB/ with names that start with AAA |
| /dirC/ | [ENG:A-Z][ENG:0-9].* | Find all files in /dirC/ with names that start with a capital and followed by a digit (A0-Z9) |

 **Note: Accepting file names from a previous step** This option allows even more flexibility in combination with other steps such as "Get File Names". You can create your file name and pass it to this step. This way the file name can come from any source; a text file, database table, and so on.

| Option | Description |
|---|--|
| Accept file names from previous steps | Enables the option to get file names from previous steps |
| Step to read file names from | Step from which to read the file names |
| Field in the input to use as file name | Text File Input looks in this step to determine which filenames to use |

Content Tab

Options under the Content tab allow you to specify the format of the text files that are being read. Below is a list of the options associated with this tab:

| Option | Description |
|--|---|
| File type | Can be either CSV or Fixed length. Based on this selection, Spoon will launch a different helper GUI when you click Get Fields in the 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. |
| Allow breaks in enclosed fields? | Not implemented |
| Escape | Specify an escape character (or characters) if you have these types of characters in your data. If you have \ as an escape character, the text 'Not the nine o'clock news' (with ' the enclosure) gets parsed as Not the nine o'clock news. |
| Header & number of header lines | Enable 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 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 and number of wraps | Use if you deal with data lines that have wrapped beyond a specific page limit; note that headers and footers are never considered wrapped |
| Paged layout and page size and doc header | 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 |
| Compression | Enable if your text file is in a Zip or GZip archive. |
| No empty rows | Do not send empty rows to the next steps. |
| Include file name in output | Enable if you want the file name to be part of the output |
| File name field name | Name of the field that contains the file name |
| Rownum in output? | Enable if you want the row number to be part of the output |
| Row number field name | Name of the field that contains the row number |
| Format | 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 searches your system for available encodings. |
| Limit | The maximum number of rows that will be read from the file |
| Be lenient when parsing dates? | Disable if you want strict parsing of data fields; if 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 such as "February 2nd, 2010;" parsing this date on a system running in the French (fr_FR) locale would not work because February is called Février in that locale. |
| Add filenames to result | Enable so that output file names are added as a field in the results |

Error Handling Tab

Options under the Error Handling tab allow you to specify how the step reacts when errors (such as, malformed records, bad enclosure strings, wrong number of fields, premature line ends), occur. The table below describes the options available for Error handling:

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

Filters Tab

Options under the Filters tab allow you to specify the lines you want to skip in the text file. The table below describes the available options for defining filters:


| Option | Description |
|------------------------|--|
| Filter string | The string for which to search |
| Filter position | The position where the filter string has to be at in the line. Zero (0) is the first position in the line. If you specify a value below zero (0) here, the filter string is searched for in the entire string. |
| Stop on filter | Specify Y here to stop processing the current text file when the filter string is encountered. |
| Positive Match | Includes the rows where the filter condition is found (include). The alternative is that those rows are avoided (exclude). |

Fields Tab


The options under the Fields tab allow you to specify the information about the name and format of the fields being read from the text file. Available options include:


| Option | Description |
|------------------|--|
| Name | Name of the field |
| Type | Type of the field can be either String, Date or Number |
| Format | See Number Formats below 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 E5.000,00 |
| 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 |

| Option | Description |
|-----------|---|
| Default | Default value in case the field in the text file was not specified (empty) |
| Trim Type | Type trim this field (left, right, both) before processing |
| Repeat | If the corresponding value in this row is empty, repeat the one from the last time it was not empty (Y/N) |

 **Note: Number formats** The information about Number formats was taken from the Sun Java API documentation, [Decimal Formats](#).

| 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'". |

 **Note: Scientific Notation** In a pattern, the exponent character immediately followed by one or more digit characters indicates scientific notation (for example, "0.###E0" formats the number 1234 as "1.234E3").

 **Note: Date formats** The information about Date formats was taken from the Sun Java API documentation, [Date Formats](#).

| 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 |


| Letter | Date or Time Component | Presentation | Examples |
|--------|------------------------|-------------------|--|
| H | Hour in day (0-23) | Number 0 | n/a |
| k | Hour in day (1-24) | Number 24 | n/a |
| K | Hour in am/pm (0-11) | Number 0 | n/a |
| h | Hour in am/pm (1-12) | Number 12 | n/a |
| m | Minute in hour | Number 30 | n/a |
| s | Second in minute | Number 55 | n/a |
| S | Millisecond | Number 978 | n/a |
| z | Time zone | General time zone | Pacific Standard Time; PST; GMT-08:00 |
| Z | Time zone | RFC 822 time zone | -0800 |

JMS Consumer

The Java Messaging Service (JMS) Consumer step allows Pentaho Data Integration to receive text messages from any JMS server. For example, you could use JMS Consumer step to define a long running transformation that updates a data warehouse every time a JMS message is received.

You must be familiar with JMS messaging to use this step. Additionally, you must have a message broker like [Apache ActiveMQ](#) available before you configure this step. If you are using the Java Naming and Directory Interface (JNDI) to connect to JMS, you must have the appropriate connection information.

JMS Consumer Options

| Option | Description |
|----------------------------|--|
| Step Name | Optionally, you can change the name of this step to fit your needs. |
| ActiveMQ Connection | Enable ActiveMQ Connection you are using ActiveMQ as your message broker. |
| JMS URL | Enter the appropriate broker URL. |
| Username | Enter the ActiveMQ user name. |
| Password | Enter the ActiveMQ password. |
| Jndi Connection | Enable JNDI Connection if you are using the Java Naming and Directory Interface (JNDI) to connect to JMS |
| Jndi URL | The URL for the JNDI connection |
| Topic/Queue | Select Topic or Queue from the drop down list to specify whether you want to use a Topic or Queue delivery model. Topic uses a publish/subscribe delivery model meaning that a one message can be delivered to multiple consumers. Messages are delivered to the topic destination, and ultimately to all active consumers who are subscribers of the topic. Also, any number of producers can send messages to a topic destination; each message can be delivered to any number of subscribers. If there are no registered consumers, the topic destination does not hold messages unless it has durable subscription for inactive consumers. A durable subscription represents a consumer registered with the topic destination that can be inactive at the time the messages are sent to the topic. Queue uses a point-to-point delivery model. In this model, a message is delivered from a single producer to a single consumer. The messages are delivered to the destination, which is a queue, and then delivered to one of the consumers registered for the queue. While any number of producers can send messages to the queue, each message is guaranteed to be delivered, and consumed by one consumer. If no consumers are registered to consume the messages, the queue holds them until a consumer registers to consume them. |
| Destination | Specify the queue or topic name. |
| Receive Timeout | Specify the time to wait for incoming messages in milliseconds.  Note: A timeout setting of zero never expires. |
| Field Name | Specify the field name that contains the contents of the message. |

JMS Producer

The Java Messaging Service (JMS) Producer step allows Pentaho Data Integration to send text messages to any JMS server. For example, you could use the JMS Producer step to define a transformation that for every update of a warehouse, posts to a JMS queue that could launch another job that flushes an application cache.

You must be familiar with JMS messaging to use this step. Additionally, you must have a message broker like [Apache ActiveMQ](#) available before you configure this step. If you are using the Java Naming and Directory Interface (JNDI) to connect to JMS, you must have the appropriate connection information.



Note: Place JMS Library jars for the ConnectionFactory and other supporting classes in the `.../data-integration/plugins/pdi-jms-plugin/lib` directory.

JMS Producer Options


| Option | Description |
|-----------------------------|---|
| Step Name | Optionally, you can change the name of this step to fit your needs. |
| ActiveMQ Connection | Enable ActiveMQ Connection you are using the message broker. |
| JMS URL | Enter the appropriate broker URL. |
| Username | Enter the ActiveMQ user name |
| Password | Enter the ActiveMQ password |
| JNDI Connection | Enable JNDI Connection if you are using the Java Naming and Directory Interface (JNDI) to connect to JMS |
| Jndi URL | The URL for the JNDI connection |
| Topic/Queue | Select Topic or Queue from the drop down list to specify whether you want to use a Topic or Queue delivery model. Topic uses a publish/subscribe delivery model meaning that a one message can be delivered to multiple consumers. Messages are delivered to the topic destination, and ultimately to all active consumers who are subscribers of the topic. Also, any number of producers can send messages to a topic destination; each message can be delivered to any number of subscribers. If there are no registered consumers, the topic destination does not hold messages unless it has durable subscription for inactive consumers. A durable subscription represents a consumer registered with the topic destination that can be inactive at the time the messages are sent to the topic. Queue uses a point-to-point delivery model. In this model, a message is delivered from a single producer to a single consumer. The messages are delivered to the destination, which is a queue, and then delivered to one of the consumers registered for the queue. While any number of producers can send messages to the queue, each message is guaranteed to be delivered, and consumed by a single consumer. If there are no consumers registered to consume the messages, the messages are held in the queue until a consumer registers to consume them. |
| Destination | Specify the queue or topic name. |
| Header Properties | If the header properties file is specified, then the name/value pairs are submitted with the message text as JMS string properties. |
| Is field a filename? | Enable if the message is based on a field name. In this instance, the contents of the file, (not the field name), are sent. |
| Field Name | If applicable, select the name of the field from the list. |

Table Output

The Table Output step allows you to load data into a database table. Table Output is equivalent to the DML operator, **INSERT**. This step provides configuration options for target table and a lot of housekeeping and/or performance-related options such as Commit Size and Use batch update for inserts.

If you have a Postgres or MySQL database that has identity columns and you are inserting a record, as part of the insert, the JDBC driver will typically return the auto-generated key it used when performing the insert.

Table Output Options

| Option | Description |
|---|--|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Connection | The database connection to which data is written (see Managing Connections) |
| 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 periods in them. |
| Target table | The name of the table to which data is written. |
| Commit size | Use transactions to insert rows in the database table. Commit the connection every N rows if N is larger than zero (0); otherwise, don't use transactions. (Slower)  Note: Transactions are not supported on all database platforms. |
| Truncate table | Select if you want the table to be truncated before the first row is inserted into the table |
| Ignore insert errors | Makes PDI 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. |
| Partition data over tables | Used 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: <pre>CREATE OR REPLACE VIEW SALES AS SELECT * FROM SALES_200501 UNION ALL SELECT * FROM SALES_200502 UNION ALL SELECT * FROM SALES_200503 UNION ALL SELECT * FROM SALES_200504</pre> |
| Use batch update for inserts | Enable if you want to use batch inserts. This feature groups inserts statements to limit round trips to the database. This is the fastest option and is enabled by default. |
| 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 | Enable if you want to get back the key that was generated by inserting a row into the table |


| Option | Description |
|----------------------------------|---|
| Name of auto-generated key field | Specifies the name of the new field in the output rows that contains the auto-generated key |
| SQL | Generates the SQL to create the output table automatically |

Text File Output

The Text File Output step is used to export data to text file format. This is commonly used to generate Comma Separated Values (CSV files) that can be read by spreadsheet applications. It is also possible to generate fixed width files by setting lengths on the fields in the fields tab.

File Tab


The options under the File tab is where you define basic properties about the file being created, such as:

| Option | Description |
|------------------------------------|---|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Filename | Specify the CSV file from which to write |
| Run this as a command instead? | Enable to "pipe" the results into the command or script you specify |
| Accept file name from field? | Enable to specify the file name(s) in a field in the input stream |
| File name field | When the previous option is enabled, you can specify the field that will contain the filename(s) at runtime. |
| Extension | Adds a point and the extension to the end of the file name. (.txt) |
| Include stepnr in filename | If you run the step in multiple copies (Launching several copies of a step), the copy number is included in the file name, before the extension. (_0). |
| Include partition nr in file name? | Includes the data partition number in the file name |
| Include date in file name | Includes the system date in the filename (_20101231) |
| Include time in file name | Includes the system time in the filename (_235959) |
| Show file name(s) | Displays a list of the files that will be generated  Note: This is a simulation and depends on the number of rows that will go into each file. |

Content tab


The content tab contains the following options for describing the content being read:

| Option | Description |
|------------------------------------|---|
| Append | Enable to append lines to the end of the specified file |
| Separator | Specify the character that separates the fields in a single line of text; typically this is semicolon (;) or a tab |
| Enclosure | A pair of strings can enclose some fields. This allows separator characters in fields. The enclosure string is optional. |
| Force the enclosure around fields? | 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 | Can be either DOS or UNIX; UNIX files have lines are separated by line feeds, DOS files have lines separated by carriage returns and line feeds |

| Option | Description |
|---------------------------------------|---|
| Compression | Specify the type of compression, .zip or .gzip to use when compressing the output.  Note: Only one file is placed in a single archive. |
| 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 searches your system for available encodings. |
| Fast data dump (no formatting) | Improves the performance when dumping large amounts of data to a text file by not including any formatting information |
| Right pad fields | Enable so that fields are padded to their defined width on the right |
| Split every ... rows | If the 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 tab

The fields tab is where you define properties for the fields being exported. The table below describes each of the options for configuring the field properties:

| Option | Description |
|----------------------|---|
| Name | The name of the field |
| Type | Type of the field can be either String, Date or Number. |
| Format | The format mask to convert with. See Number Formats for a complete description of format symbols. |
| Length | The length option depends on the field type follows: <ul style="list-style-type: none"> • Number - Total number of significant figures in a number • String - total length of string • Date - length of printed output of the string (for example, 4 returns year) |
| Precision | The precision option depends on the field type as follows: <ul style="list-style-type: none"> • Number - Number of floating point digits • String - unused • Date - unused |
| Currency | Symbol used to represent currencies like \$10,000.00 or E5.000,00 |
| Decimal | A decimal point can be a "." (10,000.00) or "," (5.000,00) |
| Group | A grouping can be a "," (10,000.00) or "." (5.000,00) |
| Trim type | Type trim this field (left, right, both) before processing  Note: Trimming works when there is no field length given only. |
| Null | If the value of the field is null, insert this string into the text file |
| Get | Click to retrieve the list of fields from the input fields stream(s) |
| Minimal width | Change 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, you write 1, and so on. String fields will no longer be padded to their specified length. |

Select Values


The Select Values step is useful for selecting, removing, renaming, changing data types and configuring the length and precision of the fields on the stream. These operations are organized into different categories:

- **Select and 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 metadata) of one or more fields

An example of a transformation that includes this step is located at `samples/transformations/Select values - some variants.ktr` and `samples/transformations/Select Values - copy field values to new fields.ktr`

Select & Alter Options


This tab contains options for selecting and changing data types and fields. The **Get Fields to Select** button will retrieve available fields based on the existing input steps and populate the entries in this tab. Click **Edit Mapping** to open a mapping dialog to easily define multiple mappings between source and target fields.

 **Note:** Edit Mapping will only work if there is only one target output step.

| Option | Description |
|--|---|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Fieldname | Click to insert fields from all input steams to the step |
| Rename to | Click to insert fields from all input steams to the step |
| Length | Enable if you want to implicitly select all other fields from the input stream(s) that are not explicitly selected in the Fields section. |
| Precision | The precision option depends on the field type, but only Number is supported; it returns the number of floating point digits. |
| Include unspecified fields, ordered by name | Enable if you want to implicitly select all other fields from the input stream(s) that are not explicitly selected in the Fields section |

Remove

Simply name the fields from the inputstream that you want to remove.

 **Note:** Field removals are slow because of the nature of the queries that they generate.

Meta-data

Options under this tab allow you to rename, change data types, and change the length and precision of fields coming into the Select Values step. Click **Get fields to change** to import fields from previous steps. A lot of data type conversions are also possible with this tab.

| Option | Description |
|------------------|--|
| Fieldname | The name of the imported field. |
| Rename to | If you want to rename this field, this is where you put the new name. |
| Type | The data type for this field. |
| Length | The field length. |
| Precision | The precision option depends on the field type, but only Number is supported; it returns the number of floating point digits. |

| Option | Description |
|----------------------|---|
| Binary to Normal? | Converts a string to a numeric data type, when appropriate. |
| Format | The format mask (number type or date format). |
| Encoding | Specifies 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 searches your system for available encodings and populates this list accordingly. |
| Decimal | A decimal point; this is either a dot or a comma. |
| Grouping | A method of separating units of thousands in numbers of four digits or larger. This is either a dot or a comma. |
| Currency | Symbol used to represent currencies. |
| Date Format Lenient? | Determines whether the date parser is strict or lenient. Leniency means that invalid date values are processed. If set to N , only strictly valid date values will be accepted; if set to Y , the parser will attempt to determine the intention of an incorrect date, if possible, and correct it. |

Dummy (do nothing)

The Dummy step does not do anything. Its primary function is to be a placeholder for testing purposes. For example, to have a transformation, you must have at least two steps connected to each other. If you want to test a file input step, you can connect it to a dummy step.

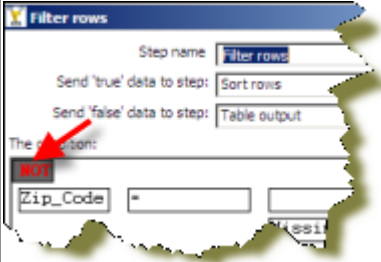
Filter Rows


The Filter Rows step allows you to filter rows based on conditions and comparisons. Once this step is connected to a previous step (one or more and receiving input), you can click on the "<field>", "=" and "<value>" areas to construct a condition.



Note: To enter an IN LIST operator, use a string value separated by semicolons. This also works on numeric values like integers. The list of values must be entered with a string type, e.g.: 2;3;7;8

Filter Row Options

| Option | Description |
|---------------------------|--|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Send 'true' data to step | The rows for which the condition specified is true are sent to this step |
| Send 'false' data to step | The rows for which the condition specified are false are sent to this step |
| The Condition | Click the NOT to negate the condition.  Click <Field> to select from a list of fields from the input stream(s) to build your condition(s). |

| Option | Description |
|----------------------|--|
| | Click <value> to enter a specific value into your condition(s). To delete a condition, right-click and select Delete Condition . |
| Add Condition | Click  (Add condition) to add conditions. Add condition converts the original condition into a sub-level condition. Click a sub-condition to edit it by going down one level in the condition tree. |

Filtering Rows Based on Values from Variables

The filter rows step detects only fields in the input stream. If you want to filter rows based in a variable value, you must modify the previous step (a table input for example) and include the variable as another field, such as:

```
${myvar}=5
```

A query:

```
SELECT field1,
field2,
${myvar} AS field3
FROM table
WHERE field1=xxxx
```

Then in the filter row condition, you can have the following...

```
field1 = field3
```

Alternatively, you can use the simple **Get Variables** step to set parameters in fields.


Database Lookup

The Database Lookup step allows you to look up values in a database table. Lookup values are added as new fields onto the stream.

Database Lookup Options

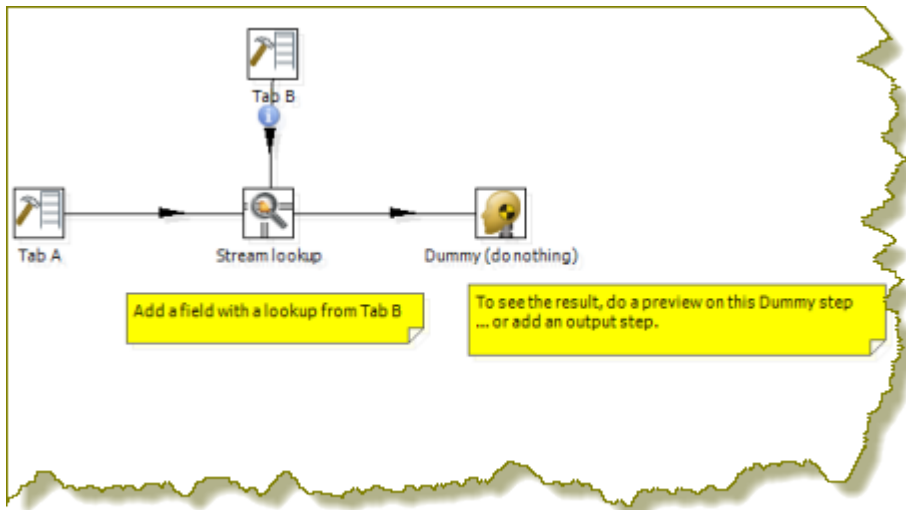
| Option | Description |
|--|--|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Connection | Select the database connection for the lookup |
| Lookup schema | Specify the target schema to use for the lookup |
| Lookup table | Specifies the name of the table used in the lookup process |
| Enable cache? | This option caches database lookups. This means that the database is expected to return the same value all the time for a certain lookup value. |
| Cache size in rows | Specify the size of the cache to use in rows. |
| Load all data from table | Pre-loads the cache with all the data in the lookup table. This can improve performance by lowering lookup latency; however, if you have a large table you may run out of memory. |
| Keys to look up table | Specify the keys necessary to perform the lookup. |
| Do not pass the row if the lookup fails | Enable to avoid passing a row when lookup fails |
| Fail on multiple results? | Enable to force the step to fail if the lookup returns multiple results. |
| Order by | If the lookup query returns multiple results, the ORDER BY clause helps you to select the record to take. For example, ORDER BY would allow you to pick the customer with the highest sales volume in a specified state. |

| Option | Description |
|--------------------------|--|
| Get Fields | Click to return a list of available fields from the input stream(s) of the step |
| Get lookup fields | Click to return a list of available fields from the lookup table that can be added to the step's output stream |

 **Important:** ! If other processes are changing values in the table where you perform a lookup, do not cache values. In all other instances, caching values increases the performance substantially because database lookups are relatively slow. If you can't use cache, consider launching several copies of the simultaneously. A simultaneous launch keeps the database busy using different connections.

Stream Lookup

The Stream Lookup 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.




Stream Lookup Options

| Option | Description |
|--|---|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Lookup step | The step name where the lookup data is coming from |
| The keys to lookup... | Allows you to specify the names of the fields that are used to look up values. Values are always searched using the "equal" comparison |
| Preserve memory | Encodes rows of data to preserve memory while sorting |
| Specify the fields to retrieve | Specifies the fields to retrieve on a successful lookup |
| Key and value are exactly one integer field | Preserves memory while executing a sort |
| Use sorted list | Enable to store values using a sorted list; this provides better memory usage when working with data sets containing wide rows |
| Get fields | Automatically fills in the names of all the available fields on the source side; you can then delete all the fields you don't want to use for lookup. |
| Get lookup fields | Automatically inserts the names of all the available fields on the lookup side. You can then delete the fields you don't want to retrieve |

Web Services Lookup

Use the Web Services Lookup step to perform a Web Services lookup using the Web Services Description Language (WSDL). This step has limitations as described below:

- Only SOAP WSDL requests / responses are understood. The other variations of the WSDL standard are not yet implemented.
- Not all WSDL XML dialects and formats are as easily read as we would like. In those cases, you need to specify (manually) what the input and output fields look like.
- Data conversion is performed within the step. In instance where you have dates and numbers you may encounter errors. If you encounter conversion errors return Strings and convert them in a **Select Values** step. See [Select Values](#).

| Option | Description |
|------------------------------------|---|
| Step Name | Optionally, you can change the name of this step to fit your needs. |
| URL | The base URL pointing to the WSDL document that will be retrieved |
| Load | Loads the WSDL at the specified URL and tries to populate the input and output tabs and fields automatically  Note: If this does not work, you can still try to manually specify the input and output fields using the Add Input and Add Output buttons. |
| The number of rows per call | The number of rows to send with each WSDL call |
| Pass input data to output | If disabled, the input will be discarded and only the WSDL output will be passed along to the next steps |
| v2.x/3.x compatibility mode | Version 2.0 engine was kept to make sure older steps would still work correctly |
| Repeating element name | The name of the repeating element in the output XML (if any). |
| HTTP authentication | The user name and password if these are required for the Web service. |
| Proxy to use | The proxy host and port information |
| Add Input / Add Output | The input and output specifications of the WSDL service |

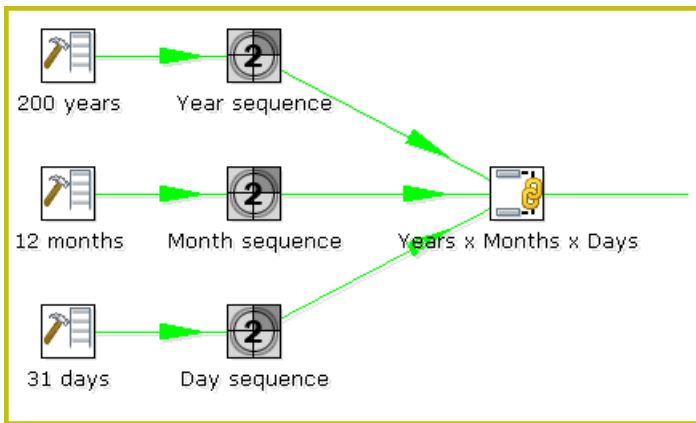
Basic Web Services - Web Services Lookup Step

In this scenario the Web service that is accessed is described with a WSDL 1.1 specification. The step can load this specification in one operation allowing the you to select and set input and output parameters. Output parameters are added to the step's output steam and passed to another step for processing. There is no need to modify the SOAP request in this scenario as the Web service does not need any information other that the parameter that is sent.

The *Web Services - NOAA Latitude and Longitude.ktr* located in the samples folder (`...\data-integration\samples\transformations`) is an example of this scenario.

Join Rows (Cartesian Product)


The Join Rows step allows combinations of all rows on the input streams (Cartesian product) as shown below:



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

The [Merge Join](#) step provides you with better performance in most cases.

Join Rows Options

| Option | Description |
|-------------------------|---|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Temp directory | Specify the name of the directory where the system stores temporary files in case you want to combine more than the cached number of rows |
| TMP-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 system reads data from temporary files; required when you want to combine large row sets that do not fit into memory |
| The Condition(s) | You can enter a complex condition to limit the number of output row.  Note: The fields in the condition must have unique names in each of the streams. |


Merge Rows

Merge rows allows you to compare two streams of rows. This is useful for comparing data from two different times. It is often used in situations where the source system of a data warehouse does not contain a date of last update.

The two streams of rows, a reference stream (the old data) and a compare stream (the new data), are merged. Only the last version of a row is passed to the next steps each time. The row is marked as follows:

- **identical** — the key was found in both streams and the values to compare are 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 when it is "deleted."

 **Note: IMPORTANT!** Both streams must be sorted on the specified key(s).

Merge Rows Options

| Option | Description |
|------------------|---|
| Step Name | Optionally, you can change the name of this step to fit your needs. |

| Option | Description |
|-----------------------|--|
| Reference rows origin | Specify the step origin for the reference rows |
| Compare rows origin | Specify the step origin for the compare rows |
| Flag fieldname | Specify the name of the flag field on the output stream |
| Keys to match | Specify fields containing the keys on which to match; click Get Key Fields to insert all of the fields originating from the reference rows step |
| Values to compare | Specify fields containing the values to compare; click Get value fields to insert all of the fields from the originating value rows step |

Combination Lookup/Update

The Combination Lookup-Update step allows you to store information in a junk-dimension table. It can sometimes be used to maintain Kimball pure Type 1 dimensions.

This step will...


- Look up combination of business key field1... fieldn from the input stream in a dimension table
- If this combination of business key fields exists, return its technical key (surrogate id)
- If this combination of business key doesn't exist yet, insert a row with the new key fields and return its (new) technical key
- 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


This step creates/maintains 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 maintain the key information only. You must update the non-key information in the dimension table; for example, by putting an update step (based on technical key) after the combination update/lookup step.


Pentaho Data Integration will store the information in a table where the primary key is the combination of the business key fields in the table. This process can be slow if you have a large number of fields, Pentaho Data Integration 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.

Combination Lookup/Update Options

| Option | Description |
|--------------------|--|
| Step Name | Optionally, you can change the name of this step to fit your needs. |
| Connection | Name of the database connection on which the dimension table resides. |
| Target schema | Allows you to specify a schema name to improve precision in the quoting and allow for table names with dots '.' in them. |
| Target table | Name of the dimension table. |
| Commit size | Setting this to 10 will generate a commit every 10 inserts or updates. |
| Cache size in rows | <p>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.</p> <p> Note: 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.</p> <p>A cache size of 0 caches as many rows as possible and until your JVM runs out of memory. Use this option wisely</p> |

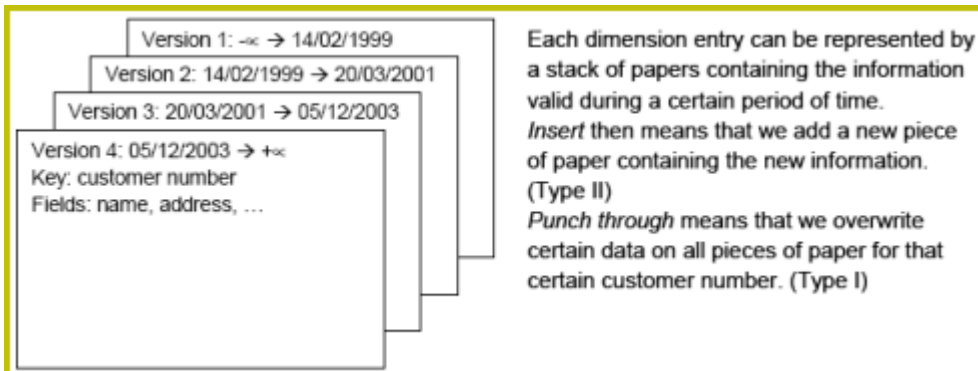
| Option | Description |
|----------------------------------|--|
| | with dimensions that can't grow too large. A cache size of -1 means that caching is disabled. |
| Key fields | Specify the names of the keys in the stream and in the dimension table. This will enable the step to do the lookup. |
| Technical key field | This indicates the primary key of the dimension. It is also referred to as Surrogate Key. |
| Creation of technical key | Specify how the technical key is generated, options that are not available for your connection are disabled: <ul style="list-style-type: none"> • 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, for example) • Use auto increment field — Use an auto increment field in the database table to generate the technical key (typical for DB2, for example.). |
| Remove lookup fields? | Enable to remove all the lookup fields from the input stream in the output. The only extra field added is then the technical key. |
| Use hashcode | Enable to use a hash code. |
| Hashcode field in table | 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. <p> Important: This hash code is NOT unique. As such it makes no sense to place a unique index on it.</p> |
| Date of last update field | When required, specify the date of last update field (timestamp) from the source system to be copied to the data warehouse. For example, when you have an address without a primary key. The field will not be part of the lookup fields (nor be part in the hash code calculation). The value is written once only because any change results in a new record being written. |
| 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. |

 **Note:** The Combination Lookup/Update step assumes that the dimension table it maintains is not updated concurrently by other transformations/applications. For example, when you use 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 keys will be cached locally, so if multiple transformations update the dimension table simultaneously you will most likely get errors on duplicate technical keys. Pentaho recommends that you do not concurrently update a dimension table, even if you are using a database sequence or auto increment technical key, because of possible conflicts between transformations.

 **Note:** 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.

Dimension Lookup/Update


The Dimension Lookup/Update step 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 step to update a dimension table, it may also be used to look up values in a dimension.



In this dimension implementation each entry in the dimension table has the following properties:

| Option | Description |
|----------------------------|---|
| Technical key | This is the primary key of the dimension. |
| Version field | Shows the version of the dimension entry (a revision number). |
| Start of date range | This is the field name containing the validity starting date. |
| End of date range | This is the field name containing the validity ending date. |
| Keys | These are the keys used in your source systems. For example: customer numbers, product id, etc. |
| 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. If you have "NOT NULL" fields in your table, adding an empty row causes the entire step to fail! Make sure that you have a record with the ID field = 0 or 1 in your table if you do not want PDI to insert a potentially invalid empty record.


In version 3.2.0, a number of optional fields (in the "Fields" tab) that are automatically managed by the step were added. You can specify the table field name in the "Dimension Field" column. These are the optional fields:

- Date of last insert or update (without stream field as source): Adds and manages a Date field
- Date of last insert (without stream field as source): Adds and manages a Date field
- Date of last update (without stream field as source): Adds and manages a Date field
- Last version (without stream field as source): Adds and manages a Boolean field. (converted into Char(1) or boolean database data type depending on your database connection settings and availability of such data type)


Lookup

In read-only mode (update option is disabled), the step only performs lookups in a slowly changing dimension. The step will perform a lookup in the dimension table on the specified database connection and in the specified schema. To perform the lookup it uses not only the specified natural keys (with an "equals" condition) but also the specified "Stream datefield" (see below). The condition that is applied is: "Start or table date range" >= "Stream datefield" AND "End or table date range" < "Stream datefield"

In the event no "Stream datefield" is specified the step uses the current system date to find the correct dimension version record.

 **Note:** If you use an "alternative start date" (Since version 3.2) the SQL clause described above will differ slightly.

In the event that no row is found, the "unknown" key is returned. This will be 0 or 1 depending on whether you selected an auto-increment field for the technical key field). Please note that we don't make a difference between "Unknown", "Not found", "Empty", "Illegal format", etc. These nuances can be added manually however. Nothing prevents you from flushing out these types before the data hits this step with a Filter, regular expression, etc. We suggest you manually add values -1, -2, -3, etc for these special dimension entry cases, just like you would add the specific details of the "Unknown" row prior to population of the dimension table.


 **Important:** Because SQL is used to look up the technical key in the dimension table, take the following precautions:

- Do *not* use NULL values for your natural key(s). Null values cannot be compared and are not indexed by most databases.
- Be aware of data conversion issues that occur if you have data types in your input streams that are different from the data types in your natural key(s). If you have Strings in the steps input and in the database you use an Integer for example, make sure you are capable of converting the String to number. See it as a best practice to do this before this step to make sure it works as planned. Another typical example of problems is with floating point number comparisons. Pentaho recommends you use data types such as Integer or Long Integers. Do not use Double, Decimal or catch-all data types such as Oracle's Number (without length or precision; it implicitly uses precision 38 causing the use of the slower BigInteger data type)

Update




In update mode (update option is enabled) the step first performs a lookup of the dimension entry as described in the "Lookup" section above. The result of the lookup is different though. Not only the technical key is retrieved from the query, but also the dimension attribute fields. A field-by-field comparison then follows. Results are as follows:


- The record was not found, new record is inserted into the table.
- The record was found and any of the following is true:
 - One or more attributes were different and had an "Insert" (Kimball Type II) setting: A new dimension record version is inserted
 - One or more attributes were different and had a "Punch through" (Kimball Type I) setting: These attributes in all the dimension record versions are updated
 - One or more attributes were different and had a "Punch through" (Kimball IType I) setting: These attributes are updated in all the dimension record versions
 - One or more attributes were different and had an "Update" setting: These attributes in the last dimension record version are updated
- All the attributes (fields) were identical : No updates or insertions are performed

 **Note:** If you mix Insert, Punch Through and Update options in this step, this algorithm acts like a Hybrid Slowly Changing Dimension. (it is no longer just Type I or II, it is a combination)

The following table provides a more detailed description of the options for the Dimension Lookup/Update step:

| Option | Description |
|------------------------------|---|
| Step name | Optionally, you can change the name of this step to fit your needs |
| Update the dimension? | Enable to update the dimension based on the information in the input stream; if not enabled, the dimension only performs lookups and adds the technical key field to the streams |
| Connection | Name of the database connection on which the dimension table resides |
| Target schema | Schema name to improve precision in the quoting and allow for table names that contain dots '.' |
| Target table | Name of the dimension table |
| Commit size | Setting commit size to 10 generates a commit every 10 inserts or updates |
| Caching | <ul style="list-style-type: none"> • Enable the cache? Enable data caching in this step; since version 3.2, set a cache size of ≥ 0 in previous versions or -1 to disable caching |

| Option | Description |
|----------------------------------|---|
| | <ul style="list-style-type: none"> • Pre-load cache? Since version 3.2 allows you to read the complete contents of a dimension table prior to performing lookups (updates are not yet supported) with the sole intention of speeding up lookup performance. Performance is increased by the elimination of the round trips to the database and by the sorted list lookup algorithm. • Cache size in rows 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. <p> Note: Only the last version of a dimension entry is kept in memory (unless pre-load is enabled). 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.</p> <p> Important: 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 grow too large. A cache size of -1 means that caching is disabled.</p> |
| Keys tab | The names of the keys in the stream and in the dimension table; enables the step to perform the lookup |
| Fields tab | For each of the fields you must 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. |
| Technical key field | <p>The primary key of the dimension; also referred to as Surrogate Key. Use the new name option to 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, ...</p> <p> Note: Renaming technical keys is only possible during lookup mode, not when running in update.</p> |
| Creation of technical key | <p>Indicates how the technical key is generated; options that are not available for your connection are grayed out</p> <ul style="list-style-type: none"> • 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 | The name of the field in which to store the version (revision number) |

| Option | Description |
|---------------------------------------|--|
| Stream Datefield | 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 do not have such a date, the system date is used. When the dimension entries are looked up (Update the dimension is not selected) the date field entered into the stream datefield is used to select the appropriate dimension version based on the date from and date to dates in the dimension record. |
| Date range start field | Specify the names of the dimension entries start range. |
| Use an alternative start date? | When enabled, you can choose an alternative to the "Min. Year"/01/01 00:00:00 date that is used. You can use any of the following: <ul style="list-style-type: none"> • System date Use the system date as a variable date/time • Start date of transformation Use the system date, taken at start of the transformation for the start date • Empty (null) value • Column value Select a column from which to take the value <p> Important: It is possible to create a non-conformed dimension with these options; use them wisely, however. Not all possibilities make sense!</p> |
| Table daterange end | The names of the dimension entries end range |
| 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. |

Group By

This step allows you to calculate values over a defined group of fields. Examples of common use cases are: calculate the average sales per product or get the number of yellow shirts that we have in stock. Sample transformations that include this step are located at:

- `.../samples/transformations/Group By - Calculate standard deviation.ktr`
- `.../samples/transformations/Group by - include all rows and calculations .ktr`
- `.../samples/transformations/Group By - include all rows without a grouping.ktr`

Group By Options

| Option | Description |
|---|--|
| Step name | Optionally, you can change the name of this step to fit your needs. |
| Include all rows? | Enable if you want all rows in the output, not just the aggregation; to differentiate between the two types of rows in the output, a flag is required in the output. You must specify the name of the flag field in that case (the type is boolean). |
| Temporary files directory | The directory in which the temporary files are stored if 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 | Enable to add a line number that restarts at 1 in each group |
| Line number field name | Enable to add a line number that restarts at 1 in each group |

| Option | Description |
|-------------------------------|--|
| Always give back a row | If you enable this option, the Group By step will always give back a result row, even if there is no input row. This can be useful if you want to count the number of rows. Without this option you would never get a count of zero (0). |
| Group fields table | Specify the fields over which you want to group. Click Get Fields to add all fields from the input stream(s). |
| Aggregates table | Specify the fields that must be aggregated, the method and the name of the resulting new field. Here are the available aggregation methods: <ul style="list-style-type: none"> • Sum • Average (Mean) • Minimum • Maximum • Number of values (N) • Concatenate strings separated by , (comma) • First non-null value • Last non-null value • First value (including null) • Last value (including null) • Cumulative sum (all rows option only!) • Cumulative average (all rows option only!) • Standard deviation • Concatenate strings separated by <Value>: specify the separator in the Value column |

Modified JavaScript Value

The Modified JavaScript Value step provides a user interface for building JavaScript expressions. This step also allows you to create multiple scripts for each step. For more information about this step see [Modified JavaScript Value](#) in the Pentaho Wiki.

JavaScript Functions

| Function | Description |
|---------------------------------|---|
| 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 |
| Transform 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. |

JavaScript

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 want to insert or by dragging the object onto the Java Script panel.

Fields

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

Buttons

Get Variables

Retrieves a list of variables from your script.

Test script

Use to test the syntax of your script.

JavaScript Internal API Objects

You can use the following internal API objects (for reference see the classes in the source):

| Object | Description |
|-----------------------------------|--|
| <code>_TransformationName_</code> | A string with the actual transformation name |
| <code>_step_</code> | The actual step instance of <code>org.pentaho.di.trans.steps.scriptvalues_mod.ScriptValuesMod</code> |
| <code>rowMeta</code> | The actual instance of <code>org.pentaho.di.core.row.RowMeta</code> |
| <code>row</code> | The actual instance of the actual data Object[] |

Advanced Web Services – Modified Java Script Value and HTTP Post Steps

There are times when the SOAP message generated by the **Web Services Lookup** step is insufficient. Many Web Services require security credentials that must be placed in the SOAP request headers. There may also be a need to parse the response XML to get more information than the response values such as namespaces.

This approach uses a **Modified Java Script Value** step. You can create the SOAP envelope as needed. The step is then hopped to an **HTTP Post** step that accepts the SOAP request through the input stream and posts it to the Web Services. This is then hopped to another **Modified Java Script Value** step that is used to parse the response from the Web service.

The *General - Annotated SOAP Web Service call.ktr* in the PDI 4.1 samples folder (`... \data-integration \samples \transformations`) illustrates the use of this approach.


Hadoop File Input

The Hadoop File Input step is used to read data from a variety of different text-file types stored on a Hadoop cluster. The most commonly used formats include Comma Separated Values (CSV files) generated by spreadsheets and fixed width flat files.

This step provides you with the ability to specify a list of files to read, or a list of directories with wild cards in the form of regular expressions. In addition, you can accept file names from a previous step making file name handling more even more generic.

Below are tables that describe all available Hadoop File Input options.

File Tab Options

| Option | Description |
|--------------------|--|
| Step Name | Optionally, you can change the name of this step to fit your needs.  Note: Every step in a transformation must have a unique name. |
| File or Directory | Specifies the location and/or name of the text file to read from. Click Browse to navigate to the file (select Hadoop in the file dialogue to enter in your Hadoop credentials), and click Add 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. For example, you want to process all files that have a .txt output. (See below) |

| Option | Description |
|--|---|
| Selected Files | This table contains a list of selected files (or wild card selections) along with a property specifying if file is required or not. If a file is required and it isn't found, an error is generated. Otherwise, the file name is skipped. |
| Show filenames(s)... | Displays a list of all files that will be loaded based on the current selected file definitions. |
| Show file content | Displays the raw content of the selected file. |
| Show content from first data line | Displays the content from the first data line only for the selected file. |

Selecting file using Regular Expressions... The Text File Input step can search for files by wildcard in the form of a regular expression. Regular expressions are more sophisticated than using '*' and '?' wildcards. Below are a few examples of regular expressions:

| File Name | Regular Expression | Files selected |
|-----------|----------------------|---|
| /dirA/ | .userdata.\.txt | Find all files in /dirA/ with names containing userdata and ending with .txt |
| /dirB/ | AAA.* | Find all files in /dirB/ with names that start with AAA |
| /dirC/ | [ENG:A-Z][ENG:0-9].* | Find all files in /dirC/ with names that start with a capital and followed by a digit (A0-Z9) |

Accepting file names from a previous step... This option allows even more flexibility in combination with other steps such as "Get File Names". You can create your file name and pass it to this step. This way the file name can come from any source; a text file, database table, and so on.

| Option | Description |
|---|--|
| Accept file names from previous steps | Enables the option to get file names from previous steps |
| Step to read file names from | Step from which to read the file names |
| Field in the input to use as file name | Text File Input looks in this step to determine which filenames to use |

Content Tab

Options under the Content tab allow you to specify the format of the text files that are being read. Below is a list of the options associated with this tab:

| Option | Description |
|--|--|
| File type | Can be either CSV or Fixed length. Based on this selection, Spoon will launch a different helper GUI when you click Get Fields in the 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 enclosure allow text line 'Not the nine o'clock news.'. With ' the enclosure string, this gets parsed as Not the nine o'clock news. |
| Allow breaks in enclosed fields? | Not implemented |
| Escape | Specify an escape character (or characters) if you have these types of characters in your data. If you have \ as an escape character, the text 'Not the nine o'clock news' (with ' the enclosure) gets parsed as Not the nine o'clock news. |
| Header & number of header lines | Enable if your text file has a header row (first lines in the file); you can specify the number of times the header lines appears. |

| Option | Description |
|--|--|
| Footer & number of footer lines | Enable 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 and number of wraps | Use if you deal with data lines that have wrapped beyond a specific page limit; note that headers and footers are never considered wrapped |
| Paged layout and page size and doc header | 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 |
| Compression | Enable if your text file is in a Zip or GZip archive. Note: At the moment, only the first file in the archive is read. |
| No empty rows | Do not send empty rows to the next steps. |
| Include file name in output | Enable if you want the file name to be part of the output |
| File name field name | Name of the field that contains the file name |
| Rownum in output? | Enable if you want the row number to be part of the output |
| Row number field name | Name of the field that contains the row number |
| Format | 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 searches your system for available encodings. |
| Be lenient when parsing dates? | Disable if you want strict parsing of data fields; if 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 such as "February 2nd, 2006;" parsing this date on a system running in the French (fr_FR) locale would not work because February is called Février in that locale. |

Error Handling Tab

Options under the Error Handling tab allow you to specify how the step reacts when errors (such as, malformed records, bad enclosure strings, wrong number of fields, premature line ends), occur. The table below describes the options available for Error handling:

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

Filters Tab

Options under the Filters tab allow you to specify the lines you want to skip in the text file. The table below describes the available options for defining filters:

| Option | Description |
|------------------------|--|
| Filter string | The string for which to search |
| Filter position | The position where the filter string has to be at in the line. Zero (0) is the first position in the line. If you specify a value below zero (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. |

Fields Tab

The options under the Fields tab allow you to specify the information about the name and format of the fields being read from the text file. Available options include:

| Option | Description |
|------------------|--|
| Name | Name of the field |
| Type | Type of the field can be either String, Date or Number |
| Format | See Number Formats below 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 E5.000,00 |
| 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 | 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 | If the corresponding value in this row is empty, repeat the one from the last time it was not empty (Y/N) |

Number formats... The information about Number formats was taken from the Sun Java API documentation, [Decimal Formats](#).

| 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 |

| Symbol | Location | Localized | Meaning |
|----------|------------------|-----------|--|
| (\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 (for example, "0.###E0" formats the number 1234 as "1.234E3").

Date formats... The information about Date formats was taken from the Sun Java API documentation, [Date Formats](#).

| 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 | n/a |
| k | Hour in day (1-24) | Number 24 | n/a |
| K | Hour in am/pm (0-11) | Number 0 | n/a |
| h | Hour in am/pm (1-12) | Number 12 | n/a |
| m | Minute in hour | Number 30 | n/a |
| s | Second in minute | Number 55 | n/a |
| S | Millisecond | Number 978 | n/a |
| z | Time zone | General time zone | Pacific Standard Time; PST; GMT-08:00 |
| Z | Time zone | RFC 822 time zone | -0800 |

Hadoop File Output



The Hadoop File Output step is used to export data to text files stored on a Hadoop cluster. This is commonly used to generate Comma Separated Values (CSV files) that can be read by spreadsheet applications. It is also possible to generate fixed width files by setting lengths on the fields in the fields tab.

Below are tables that describe all available Hadoop File Output options.

File Tab


The options under the File tab is where you define basic properties about the file being created, such as:

| Option | Description |
|-----------|---|
| Step name | Optionally, you can change the name of this step to fit your needs. |

| Option | Description |
|---|---|
| |  Note: Every step in a transformation must have a unique name. |
| Filename | Specifies the location and/or name of the text file to write to. Click Browse to navigate to the file (select Hadoop in the file dialogue to enter in your Hadoop credentials) if you don't know the path and filename. |
| Extension | Adds a point and the extension to the end of the file name. (.txt) |
| Accept file name from field? | Enable to specify the file name(s) in a field in the input stream |
| File name field | When the previous option is enabled, you can specify the field that will contain the filename(s) at runtime. |
| Include stepnr in filename | If you run the step in multiple copies (Launching several copies of a step), the copy number is included in the file name, before the extension. (_0). |
| Include partition nr in file name? | Includes the data partition number in the file name |
| Include date in file name | Includes the system date in the filename (_20101231) |
| Include time in file name | Includes the system time in the filename (_235959) |
| Show file name(s) | Displays a list of the files that will be generated  Note: This is a simulation and depends on the number of rows that will go into each file. |

Content tab


The content tab contains the following options for describing the content being read:

| Option | Description |
|---|---|
| Append | Enable to append lines to the end of the specified file |
| Separator | Specify the character that separates the fields in a single line of text; typically this is semicolon (;) or a tab |
| Enclosure | A pair of strings can enclose some fields. This allows separator characters in fields. The enclosure string is optional. Enable if you want the text file to have a header row (first line in the file). |
| Force the enclosure around fields? | 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 | Can be either DOS or UNIX; UNIX files have lines separated by line feeds, 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 searches your system for available encodings. |
| Compression | Specify the type of compression, .zip or .gzip to use when compressing the output.  Note: Only one file is placed in a single archive. |
| Fast data dump (no formatting) | Improves the performance when dumping large amounts of data to a text file by not including any formatting information |
| Split every ... rows | If the number N is larger than zero, split the resulting text-file into multiple parts of N rows |

| Option | Description |
|-------------------------|--|
| Add Ending line of file | Allows you to specify an alternate ending row to the output file |

Fields tab

The fields tab is where you define properties for the fields being exported. The table below describes each of the options for configuring the field properties:

| Option | Description |
|---------------|---|
| Name | The name of the field |
| Type | Type of the field can be either String, Date or Number. |
| Format | The format mask to convert with. See Number Formats for a complete description of format symbols. |
| Length | The length option depends on the field type follows: <ul style="list-style-type: none"> Number - Total number of significant figures in a number String - total length of string Date - length of printed output of the string (for example, 4 returns year) |
| Precision | The precision option depends on the field type as follows: <ul style="list-style-type: none"> Number - Number of floating point digits String - unused Date - unused |
| Currency | Symbol used to represent currencies like \$10,000.00 or E5.000,00 |
| Decimal | A decimal point can be a "." (10,000.00) or "," (5.000,00) |
| Group | A grouping can be a "," (10,000.00) or "." (5.000,00) |
| Trim type | The trimming method to apply on the string  Note: Trimming works when there is no field length given only. |
| Null | If the value of the field is null, insert this string into the text file |
| Get | Click to retrieve the list of fields from the input fields stream(s) |
| Minimal width | Change 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, you write 1, and so on. String fields will no longer be padded to their specified length. |

Map/Reduce Input

This step defines the key/value pairs for Hadoop input. The output of this step is appropriate for whatever data integration transformation tasks you need to perform.

| Option | Definition |
|-------------|--|
| Step name | The name of this step as it appears in the transformation workspace. |
| Key field | The Hadoop input field and data type that represents the key in map/reduce terms. |
| Value field | The Hadoop input field and data type that represents the value in map/reduce terms. |


Map/Reduce Output

This step defines the key/value pairs for Hadoop output. The output of this step will become the output to Hadoop, which changes depending on what the transformation is used for.

If this step is included in a transformation used as a **mapper** and there is a combiner and/or reducer configured, the output will become the input pairs for the combiner and/or reducer. If there are no combiner or reducers configured, the output will end up written to HDFS in the output folder of the job for which it was run.

If this step is included in a transformation used as a **combiner** and there is a reducer configured, the output will become the input pairs for the reducer. If no reducer configured, the output will end up written to HDFS in the output folder of the job for which it was run.

If this step is included in a transformation used as a **reducer**, then the output will be written to HDFS in the output folder of the job for which it was run.

 **Note:** You are not able to define the data type for the key or value here; it is defined earlier in your transformation. However, a reducer or combiner that takes this output as its input will have to know what the key and value data types are, so you may need to make note of them somehow.

| Option | Definition |
|-------------|---|
| Step name | The name of this step as it appears in the transformation workspace. |
| Key field | The Hadoop output field that represents the key in map/reduce terms. |
| Value field | The Hadoop output field that represents the value in map/reduce terms. |

S3 File Output

This step exports data to a text file on an Amazon Simple Storage Service (S3) account.

File Tab

The File tab defines basic file properties for this step's output.

| Option | Description |
|------------------------------------|--|
| Step name | The name of this step in the transformation workspace. |
| Filename | The name of the output text file. |
| Accept file name from field? | When checked, enables you to specify file names in a field in the input stream. |
| File name field | When the Accept file name from field option is checked, specify the field that will contain the filenames. |
| Extension | The three-letter file extension to append to the file name. |
| Include stepnr in filename | If you run the step in multiple copies (launching several copies of a step), the copy number is included in the file name, before the extension. (_0). |
| Include partition nr in file name? | Includes the data partition number in the file name. |
| Include date in file name | Includes the system date in the filename (_20101231). |
| Include time in file name | Includes the system time (24-hour format) in the filename (_235959). |
| Show file name(s) | Displays a list of the files that will be generated. This is a simulation and |

| Option | Description |
|--------|--|
| | depends on the number of rows that will go into each file. |

Content tab

The content tab contains options for describing the file's content.

| Option | Description |
|------------------------------------|---|
| Append | When checked, appends lines to the end of the file. |
| Separator | Specifies the character that separates the fields in a single line of text; typically this is semicolon or a tab. |
| Enclosure | Optionally specifies the character that defines a block of text that is allowed to have separator characters without causing separation. Typically a single or double quote. |
| Force the enclosure around fields? | 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 | Specifies either DOS or UNIX file formats. UNIX files have lines that are separated by line feeds, DOS files have lines that are separated by carriage returns and line feeds. |
| Compression | Specifies the type of compression to use on the output file -- either zip or gzip. Only one file is placed in a single archive. |
| Encoding | Specifies 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 searches your system for available encodings. |
| Fast data dump (no formatting) | Improves the performance when dumping large amounts of data to a text file by not including any formatting information. |
| Right pad fields | When checked, fields will be right-padded to their defined width. |
| Split every ... rows | If the number N is larger than zero, splits the resulting text file into multiple parts of N rows. |
| Add Ending line of file | Enables you to specify an alternate ending row to the output file. |

Fields tab

The Fields tab defines properties for the exported fields.

| Option | Description |
|---------------|--|
| Name | The name of the field. |
| Type | The field's data type; String, Date or Number. |
| Format | The format mask (number type). |
| Length | The length option depends on the field type. Number : total number of significant figures in a number; String : total length of a string; Date : determines how much of the date string is printed or recorded. |
| Precision | The precision option depends on the field type, but only Number is supported; it returns the number of floating point digits. |
| Currency | Symbol used to represent currencies. |
| Decimal | A decimal point; this is either a dot or a comma. |
| Group | A method of separating units of thousands in numbers of four digits or larger. This is either a dot or a comma. |
| Trim type | Truncates the field (left, right, both) before processing. Useful for fields that have no static length. |
| Null | Inserts the specified string into the text file if the field value is null. |
| Get | Retrieves a list of fields from the input stream. |
| Minimal width | Minimizes field width by removing unnecessary characters (such as superfluous zeros and spaces). If set, string fields will no longer be padded to their specified length. |

RSS Input

This step imports data from an RSS or Atom feed. RSS versions 1.0, 2.0, 0.91, 0.92, and Atom versions 0.3 and 1.0 are supported.

General Tab

The General tab defines which RSS/Atom URLs you want to use, and optionally which fields contain the URLs.

| Option | Description |
|---------------------------|---|
| Step name | The name of this step in the transformation workspace. |
| URL is defined in a field | If checked, you must specify which field to retrieve the URL from. |
| URL Field | If the previous option is checked, this is where you specify the URL field. |
| URL list | A list of RSS/Atom URLs you want to pull article data from. |

Content tab

The content tab contains options for limiting input and changing output.

| Option | Description |
|---------------------------|---|
| Read articles from | Specifies a date in yyyy-MM-dd HH:mm:ss format. Only articles published after this date will be read. |
| Max number of articles | Specifies a static number of articles to retrieve, starting at the oldest. |
| Include URL in output? | If checked, specify a field name to pass the URL to. |
| Include rownum in output? | If checked, specify a field name to pass the row number to. |

Fields tab


The Fields tab defines properties for the exported fields.

| Option | Description |
|-----------|--|
| Name | The name of the field. |
| Type | The field's data type; String, Date or Number. |
| Length | The length option depends on the field type. Number : total number of significant figures in a number; String : total length of a string; Date : determines how much of the date string is printed or recorded. |
| Precision | The precision option depends on the field type, but only Number is supported; it returns the number of floating point digits. |
| Trim type | Truncates the field (left, right, both) before processing. Useful for fields that have no static length. |
| Repeat | If set to Y , will repeat this value if the next field is empty. |
| Format | The format mask (number type). |
| Currency | Symbol used to represent currencies. |
| Decimal | A decimal point; this is either a dot or a comma. |
| Grouping | A method of separating units of thousands in numbers of four digits or larger. This is either a dot or a comma. |

Notes on Error Handling

When error handling is turned on for the transformation that includes this step, the full exception message, the field number on which the error occurred, and one or more of the following codes will be sent in an error row to the error stream:

- **UnknownError**: an unexpected error. Check the "Error description" field for more details.
- **XMLError**: typically this means that the specified file is not XML.
- **FileNotFound**: an HTTP 404 error.
- **UnknownHost**: means that the domain name cannot be resolved; may be caused by network outage.
- **TransferError**: any non-404 HTTP server error code (401, 403, 500, 502, etc.) can cause this.
- **BadURL**: means that the URL cannot be understood. It may be missing a protocol or use an unrecognized protocol.
- **BadRSSFormat**: typically means that the file is valid XML, but is not a supported RSS or Atom doc type.

 **Note:** To see the full stack trace from a handled error, turn on detailed logging.

HTTP Post

This step uses an HTTP POST command to submit form data via a URL.

General Tab

The General tab defines which RSS/Atom URLs you want to use, and optionally which fields contain the URLs.

| Option | Description |
|--|---|
| Step name | The name of this step in the transformation workspace. |
| URL | The Web service URL to submit to. |
| Accept URL from field? | If checked, you must specify which field to retrieve the URL from. |
| URL field name | If the previous option is checked, this is where you specify the URL field. |
| Encoding | The encoding standard for the files being accessed. |
| Request entity field | The name of the field that will contain the POST request. When enabled, the Post a file option will retrieve the file named in this field, and post the contents of that file. |
| Post a file | If a file is defined in the Request entity field , its contents will be posted if this option is checked. |
| Result fieldname | The field that you want to post the result output to. |
| HTTP status code fieldname | The field that you want to post the status code output to. |
| Response time (milliseconds) fieldname | The field that you want to post the response time, in milliseconds, to. |
| HTTP login | If this form requires authentication, this field should contain the username. |
| HTTP password | If this form requires authentication, this field should contain the password that corresponds with the username. |
| Proxy host | Hostname or IP address of the proxy server, if you use one. |
| Proxy port | Port number of the proxy server, if you use one. |

Fields tab: Body (Header) Parameters

The Fields tab defines parameters for the HTTP request header and body. If you've filled in the URL and other necessary details in the General tab, you can use the **Get values** buttons to pre-populate the fields here. Body parameters are used in POST and PUT operations.

| Option | Description |
|----------------|--|
| # | The order that this parameter will be passed to the Web application. |
| Name | The name of the field that contains the value to map to the parameter. |
| Parameter | The parameter to map the value of Name to. |
| Put in Header? | If set to Y, the parameter will be put into the request header. |

Fields tab: Query Parameters

The Fields tab defines parameters for the HTTP request header and body. If you've filled in the URL and other necessary details in the General tab, you can use the **Get values** buttons to pre-populate the fields here. Query parameters are specified in the URL and can be used in any HTTP method.

| Option | Description |
|--------|--|
| # | The order that this parameter will be passed to the Web application. |
| Name | The name of the field that contains the value to map to the parameter. |
| Value | The value to map to the parameter. |

Job Step Reference

There are over 60 job entries associated with Pentaho Data Integration. The listing below is a subset of the most commonly-used job entries. In later versions of this document, more job entries will be added to the list. Currently, the bulk of job entry-related documentation is available in the [Pentaho Wiki](#); however, Wiki documents are maintained by the open source community and are therefore not always as comprehensive or as accurate.

General

| Name | Description |
|--------------------------------|---|
| Start | Starts execution of a job |
| Dummy | Does nothing; it is an entry point |
| Job | Executes a job |
| Transformation | Executes a previously designed transformation |

Mail

| Name | Description |
|----------------------|----------------|
| Mail | Sends an email |

Conditions

| Name | Description |
|------------------------------|--|
| File Exists | Verifies if the specified file exists on the server on which Pentaho Data Integration is running |
| Table Exists | Verifies that the specified table exists in a database |

Scripting

| Name | Description |
|----------------------------|--|
| JavaScript | Calculates a boolean expression; this result can be used to determine which step will be executed next |
| Shell | Executes a shell script on the host where the job is running |
| SQL | Executes a SQL script |

File Management

| Name | Description |
|----------------------|---|
| HTTP | Gets a file from a Web server using the HTTP protocol |

File Transfer

| Name | Description |
|--------------------------------------|---|
| Get a file with FTP | Gets one or more files from an FTP server |
| Get a file with SFTP | Gets one or more files from an FTP server using the Secure FTP protocol |

Hadoop

| Name | Description |
|--|---|
| Hadoop Copy Files | Copies one or more files from one location to another, where one or both of those locations is a Hadoop cluster |
| Hadoop Job Executor | Executes a Hadoop job via a Java class |
| Hadoop Transformation Job Executor | Executes a transformation that uses Hadoop as a data source; frequently used to create PDI-based map/reduce functions in lieu of a Java class |
| Amazon EMR Job Executor | Executes a Hadoop job using an Amazon Elastic Map/Reduce account |

Start

Start defines the starting point for job execution. Every job must have one (and only one) Start. Unconditional job hops only are available from a Start job entry. The start job entry settings contain basic scheduling functionality; however, scheduling is not persistent and is only available while the device is running.

The Data Integration Server provides a more robust option for scheduling execution of jobs and transformations and is the preferred alternative to scheduling using the Start step. If you want the job to run like a daemon process, however, enable **Repeat** in the job settings dialog box.

Dummy



The **Dummy** job entry does nothing. It is just an entry point on the canvas; however, suppose you have a transformation that processes one row at a time. You have set the transformation so that it gets the initial five records and processes the additional records five at a time. The Job script must determine whether or not processing is complete. It may need to loop back over a few times. The job workflow drawing could be tough to read in this type of scenario. The Dummy job entry makes the job workflow drawing clearer for looping. Dummy performs no evaluation.

Job

Use the **Job** job entry to execute a previously defined job. Job entry jobs allow you to perform "functional decomposition." That is, you use them to break out jobs into more manageable units. For example, you would not write a data warehouse load using one job that contains 500 entries. It is better to create smaller jobs and aggregate them.

Below are the job options listed by tab name. The name of the job entry appears above every tab.

Transformation Specification

| Option | Description |
|--------------------------------------|--|
| Name of the Job Entry | The unique name of the job entry on the canvas. A job entry can be placed on the canvas several times; however it will be the same job entry |
| Job Filename | If you are not working in a repository, specify the XML file name of the transformation to start. Click  to browse through your local files. |
| Specify by Name and Directory | If you are working in the Enterprise Repository, (or database repository) specify the name of the transformation to start. Click  to browse through the repository. |
| Specify by Reference | If you specify a transformation or job by reference, you can rename or move it around in the repository. The reference (identifier) is stored, not the name and directory. |

Advanced

| Option | Description |
|---|--|
| Copy previous results to args? | The results from a previous transformation can be sent to this one using the Copy rows to result step |
| Copy previous results to parameters? | If Execute for every input row is enabled then each row is a set of command line arguments to be passed into the transformation, otherwise only the first row is used to generate the command line arguments. |
| Execute for every input row? | Implements looping; if the previous job entry returns a set of result rows, the job executes once for every row found. One row is passed to the job at every execution. For example, you can execute a job for each file found in a directory. |

| Option | Description |
|---|---|
| Remote slave server | The slave server on which to execute the job |
| Wait for the remote transformation to finish? | Enable to block until the job on the slave server has finished executing |
| Follow local abort to remote transformation | Enable to send the abort signal to the remote job if it is called locally |

Logging Settings

| Option | Description |
|--------------------------|--|
| Specify logfile? | Enable to specify a separate logging file for the execution of this transformation |
| Append logfile? | Enable to append to the logfile as opposed to creating a new one |
| Name of logfile | The directory and base name of the log file; for example C:\logs |
| Create parent folder | Create the parent folder for the log file if it does not exist |
| Extension of logfile | The file name extension; for example, log or txt |
| Include date in logfile? | Adds the system date to the log file |
| Include time in logfile? | Adds the system time to the log file |
| Loglevel | The logging level to use |

Arguments

You can pass job command line arguments to a transformation.

Parameters



You can pass parameters to a transformation.

Transformation

The **Transformation** job entry is used to execute a previously defined transformation.

Below are the transformation options listed by tab name. The name of the job entry appears above every tab.

Transformation Specification

| Option | Description |
|-------------------------------|--|
| Name of the Job Entry | The unique name of the job entry on the canvas. A job entry can be placed on the canvas several times; however it will be the same job entry |
| Transformation Filename | If you are not working in a repository, specify the XML file name of the transformation to start. Click  to browse through your local files. |
| Specify by Name and Directory | If you are working in the Enterprise Repository, (or database repository) specify the name of the transformation to start. Click  to browse through the repository. |
| Specify by Reference | If you specify a transformation or job by reference, you can rename or move it around in the repository. The reference (identifier) is stored, not the name and directory. |

Advanced

| Option | Description |
|--------------------------------------|---|
| Copy previous results to args? | The results from a previous transformation can be sent to this one using the Copy rows to result step |
| Copy previous results to parameters? | If Execute for every input row is enabled then each row is a set of command line arguments to be passed into |

| Option | Description |
|--|--|
| | the transformation, otherwise only the first row is used to generate the command line arguments. |
| Execute for every input row? | Allows a transformation to be executed once for every input row (looping) |
| Clear list of result files before execution? | Ensures that the list or result files is cleared before the transformation is started |
| Run this transformation in a clustered mode? | As described |
| Remote slave server | The slave server on which to execute the job |
| Wait for the remote transformation to finish? | Enable to block until the job on the slave server has finished executing |
| Follow local abort to remote transformation | Enable to send the abort signal to the remote job if it is called locally |

Logging Settings

By default, if you do not set logging, Pentaho Data Integration will take log entries that are being generated and create a log record inside the job. For example, suppose a job has three transformations to run and you have not set logging. The transformations will not output logging information to other files, locations, or special configuration. In this instance, the job executes and puts logging information into its master job log.

In most instances, it is acceptable for logging information to be available in the job log. For example, if you have load dimensions, you want logs for your load dimension runs to display in the job logs. If there are errors in the transformations, they will be displayed in the job logs. If, however, you want all your log information kept in one place, you must set up logging.

| Option | Description |
|---------------------------------|--|
| Specify logfile? | Enable to specify a separate logging file for the execution of this transformation |
| Append logfile? | Enable to append to the logfile as opposed to creating a new one |
| Name of logfile | The directory and base name of the log file; for example C:\logs |
| Create parent folder | Enable to create a parent folder for the log file if it does not exist |
| Extension of logfile | The file name extension; for example, log or txt |
| Include date in logfile? | Adds the system date to the log file |
| Include time in logfile? | Adds the system time to the log file |
| Loglevel | The logging level to use |

Arguments


You can pass job command line arguments to a transformation.

Parameters

You can pass parameters to a transformation.

Mail


Use the **Mail** job entry to send a text or HTML email with optional file attachments. This job entry is used at the end of a job run in most instances. It can be used to announce both a job failure or success. For example, it is not uncommon at the end of a successful load, to send an email to a distribution list announcing that the load was successful and include a log file. If there are errors, an email can be sent to alert individuals on a distribution list.

 **Important:** No email messages are sent when a job crashes during a run. If you are bound by service level agreements or quality of service agreements you may not want to use this job entry as a notification method.

The Mail job entry requires an SMTP server. You can use authentication and security as part of the connection but you must have the SMTP credentials.

You can attach files to your email messages such as error logs and regular logs. In addition, logs can be zipped into a single archive for convenience.

Addresses

| Option | Description |
|----------------------------|--|
| Job entry name | The unique name of the job entry on the canvas. A job entry can be placed on the canvas several times; however it will be the same job entry |
| Destination Address | The destination for the email; you can specify multiple addresses if you separate them with a space or comma.  Note: Do not maintain your distribution list within a job. Rather, have your email administrators set up a list so that you can send to a specified list each time you create the job. Operational functions such as Email contents, routing, people, and so on should be managed outside of Pentaho Data Integration. |
| Cc: | An identical copy of the message is also sent to all the addresses listed in the Cc: field. To enter more than one address in the Cc: field, separate them with commas. |
| BCc: | Send to a recipient whose email address does not appear in the message |
| Sender name | Name of the person sending the email |
| Sender address | Email address of the person sending the email |
| Reply to | Email address of the person to which to send a reply |
| Contact person | The name of the contact person to be placed in the email |
| Contact phone | The contact telephone number to be placed in the email |

Server

| Option | Description |
|-----------------------------------|---|
| SMTP Server | The SMTP server address |
| Port | The port on which the SMTP Server is running |
| Authentication | Enable to set authentication to the SMTP Server |
| Authentication user | SMTP user account name |
| Authentication password | SMTP user account password |
| Use Secure Authentication? | Enable to use secure authentication |
| Secure Connection Type | Select authentication type |

Email Message

| Option | Description |
|--|--|
| Include date in message? | Enable to include date in message |
| Only send comment in mail body? | If not enabled the email will contain information about the job and its execution in addition to the comment |
| Use HTML in mail body | As described |
| Encoding | Select encoding type |
| Manage Priority | Enable to manage priority |
| Subject | As described |
| Comment | As described |

Attached Files

| Option | Description |
|-------------------------------------|---|
| Attach files to message? | Enable to attach a file to your email message |
| Select file type | As described |
| Zip files to single archive? | Enable to have attachments archived in a zip file |
| Name of the zip archive | As described |

| Option | Description |
|------------|--|
| Filename | Name of a <i>single</i> image file |
| Content ID | Automatically entered |
| Image | The full path to image (used when embedding multiple images) Click Edit to edit the path; click Delete to delete the path to the image |
| Content ID | The image content ID (used when embedding multiple images) Click Edit to edit the content ID; click Delete to delete the Content ID |

File Exists

Use the **File exists** job entry to verify that a specified file exists on the server on which Pentaho Data Integration is running. You must provide the file name. Pentaho Data Integration returns a True or False value depending on whether or not the file exists.

The File Exists job entry can be an easy integration point with other systems. For example, suppose you have a three-part data warehouse load process. The first part runs in PERL. You have batch scripts that accesses data from a remote location, performs first-level row processing, and outputs the data to a specified directory. You do not want to start the job until this is done, so you put the job on a scheduler. As soon as the task is complete, the file is placed in a well-known location so that the "file exists." That is the signal that launches the job for final processing.



 **Note:** This job entry performs one check and then moves on. If you want to poll until the files appear, use the **Wait for File** or **Wait for SQL** job entries which have a polling interval parameter.

Table Exists

Use the **Table exists** job entry to verify that a specified table exists on a database. You must provide a connection and the table name. Pentaho Data Integration returns a True or False value depending on whether or not the table exists.

Suppose you have an external system that creates a summary table or yesterday's data extract. The external system may not have performed the action yet, so you set up a polling piece that waits for the staged data to arrive in the database. There is no point in processing the job until the data is available, so you can use this job entry as a semaphore that polls the database to determine whether or not the table exists.

 **Note:** This job entry performs one check and then moves on. If you want to poll until the tables appear, use the **Wait for File** or **Wait for SQL** job entries which have a polling interval parameter.

| Option | Description |
|----------------|--|
| Job entry name | The unique name of the job entry on the canvas. A job entry can be placed on the canvas several times; however it will be the same job entry |
| Connection | The connection to use |
| Schema name | The schema name if applicable to your database |
| Table name | The name of the database table to check |

JavaScript

Use the **JavaScript** job entry to calculate a boolean expression. The result can be used to determine which step will be executed next. You can use functions, procedure calls, ANDs, ampersands, ORs, EQUALs, etc. The Javascript job entry evaluates (returns?) a true or false.

The following variables are available for the expression:

| Variable | Description |
|---------------|---|
| 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) |

| Variable | Description |
|-----------------|--|
| 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 | use if Pentaho Data Integration runs on Windows (boolean) |

Shell

Use the **Shell** job entry to execute a shell script on the host where the job is running. For example, suppose you have a program that reads five data tables and creates a file in a specified format. You know the program works. Shell allows you to do portions of your work in Pentaho Data Integration but reuse the program that reads the data tables as needed.

The Shell job entry is platform agnostic; you can use a batch file, UNIX, and so on. When you use a Shell job entry, Pentaho Data Integration makes a Java call to execute a program in a specified location. The return status is provided by the operating system call. For example, in batch scripting a return value of 1 indicates that the script was successful; a return value of 0 (zero) indicates that it was unsuccessful. You can pass command line arguments and set up logging for the Shell job entry.

General

| Option | Description |
|---------------------------------------|--|
| Job entry name | The unique name of the job entry on the canvas. A job entry can be placed on the canvas several times; however it will be the same job entry |
| Insert script | Enable to insert the shell script; click on the Script tab to manually enter your script |
| Script file name | The file name of the shell script to execute |
| Working directory | The working directory for the command or script |
| Specify logfile? | Enable to specify a separate logging file for the execution of this transformation |
| Append logfile? | Enable to append to the logfile as opposed to creating a new one |
| Name of logfile | The directory and base name of the log file; for example C:\logs |
| Extension of logfile | The file name extension; for example, log or txt |
| Include date in logfile? | Adds the system date to the log file |
| Include time in logfile? | Adds the system time to the log file |
| Loglevel | Specifies the logging level for the execution of the shell |
| Copy previous results to args? | The results from a previous transformation can be sent to the shell script using Copy rows to result step |
| Execute for every input row? | Implements looping; if the previous job entry returns a set of result rows, the shell script executes 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). |
| Fields | Values to be passed into the command/script as command line arguments. (Not used if Copy previous results to args is used) |

Script

An arbitrary script that can be used as the files contents if **Insert script** is enabled

SQL

Use the **SQL** job entry to execute an SQL script. You can execute more than one SQL statement, as long as they are separated by semi-colons. The SQL job entry is flexible; you can perform procedure calls, create and analyze tables, and more. Common uses associated with the SQL job entry include truncating tables, drop index, partition loading, refreshing materialized views, disabling constraints, disabling statistics, and so on.

| Option | Description |
|----------------------------------|--|
| Job entry name | The unique name of the job entry on the canvas. A job entry can be placed on the canvas several times; however it will be the same job entry. |
| Connection | The database connection to use |
| SQL from file | Enable to use SQL script from file |
| SQL filename | Specify SQL file name |
| Send SQL as single | If enabled the entire block is sent as a single statement on the database. If not enabled, each statement (as terminated by ';') is executed individually. |
| Use variable substitution | Resolve the SQL block against Pentaho Data Integration variables |
| SQL Script | The SQL script to execute |

HTTP

Use the HTTP job entry to retrieve a file from a Web server using the HTTP protocol. This job entry could be used to access data on partner Web sites. For example, the daily data export or daily list of customers is located at a specified Web site. Also, SaaS providers may give you a URL to locate a report. You can call that URL to retrieve an Excel file or zip file that contains the data. Salesforce requires that you use SOAP APIs to retrieve data.

If HTTP traffic is too heavy in your corporate environment, you may choose to use a proxy server with HTTP authentication.

Get a File with FTP

Use the **Get a File with FTP** job entry to retrieve one or more files from an FTP server. This job entry does not "crawl" systems. It will not, for example, access a remote directory and go to other directories to find files that match a wildcard. This job retrieves files from one directory exclusively.

General

| Option | Description |
|-----------------------------------|--|
| Job entry name | The unique name of the job entry on the canvas. A job entry can be placed on the canvas several times; however it will be the same job entry |
| FTP server name/IP address | The name of the server or the IP address |
| Server Port | Port number of the FTP server |
| Username | The user name associated with FTP server account |
| Password | The password associated the FTP server account |
| Proxy host | Proxy server host name |
| Proxy port | Proxy server port number |
| Proxy username | Proxy server account user name |
| Proxy password | Proxy server account password |
| Binary mode | Enable if files must be transferred in binary mode |
| Timeout | The FTP server timeout in seconds |
| Use Active FTP connection | Enable if you are connecting to the FTP server using Active mode; you must set your firewall to accept |

| Option | Description |
|-------------------------|---|
| | connections to the port that your FTP client will open. The default is Passive mode. |
| Control Encoding | Encoding matters when retrieving file names that contain special characters. For Western Europe and the USA, ISO-8859-1 usually suffices. Select encoding that is valid for your server. |

Files

| Option | Description |
|--------------------------------------|--|
| Remote directory | The remote directory on FTP server from which files are taken |
| Wildcard (regular expression) | Regular expression when you want to select multiple files. For example: <code>.*txt\$</code> : get all text files <code>A.*[ENG:0-9].txt</code> : files starting with A, ending with a number and <code>.txt</code> |
| Remove files after retrieval | Remove the files on the FTP server, but only after all selected files have been successfully transferred |
| Move to Folder | Moves files to specified folder |
| Create Folder | Creates folder that will contain files |
| Target Directory | The directory where you want to place the retrieved files |
| Include date in filename | Adds the system date to the filename (<code>_20101231</code>) |
| Include time in filename | Adds the system time to the filename (<code>_235959</code>) |
| Specify date time format | Enable to provide your own date/time format; the default is yyyyMMdd'_'HHmmss |
| Date time format | Select date time format |
| Add date before extension | Adds date to the file name before the extension |
| Don't overwrite files | Enable to skip, rename, or fail if a file with an identical name already exists in the target directory |
| If file exists | Action to take if a file with an identical name already exists in the target directory |
| Add filenames to result | Enable to add the file name(s) read to the result of this job |

Advanced

| Option | Description |
|--------------------|---|
| Success on | Sets conditions of success |
| Limit files | Sets number of files associated with a condition of success |

Socks Proxy

| Option | Description |
|-----------------|---|
| Host | Socks Proxy host name |
| Port | Socks Proxy port number |
| Username | User name associated with the Socks Proxy account |
| Password | Password associated with the Socks Proxy account |

Get a file with SFTP

Use the **Get a file with SFTP** job entry to retrieve one or more files from an FTP server using the Secure FTP protocol.

| Option | Description |
|-----------------------------|--|
| Job entry name | The unique name of the job entry on the canvas. A job entry can be placed on the canvas several times; however it will be the same job entry |
| SFTP server name /IP | The name of the SFTP server or the IP address |
| Port | The TCP port to use; usually 22 |
| User name | The user name to log onto the SFTP server |

| Option | Description |
|--------------------------------------|---|
| Password | The password to log onto the SFTP server |
| Copy previous results to args | Enable to use the list of result files from the previous job entry (entries) instead of the static file list below |
| Remote directory | The remote directory on the SFTP server from which we retrieve the files |
| Wildcard (regular expression) | Specify a regular expression here if you want to select multiple files. For example: *txt\$: get all text files A.*[ENG:0-9].txt : get all files starting with A ending with a number and .txt |
| Remove files after retrieval? | Enable to remove the files after they have been successfully transferred |
| Target directory | The directory where you want to place the transferred files |
| Add filename to result | Enable to add the file name(s) read to the result of this job |

Hadoop Copy Files

This job entry copies files in a Hadoop cluster from one location to another.

General

| Option | Definition |
|-------------------------------|---|
| Include Subfolders | If selected, all subdirectories within the chosen directory will be copied as well |
| Destination is a file | Determines whether the destination is a file or a directory |
| Copy empty folders | If selected, will copy all directories, even if they are empty the Include Subfolders option must be selected for this option to be valid |
| Create destination folder | If selected, will create the specified destination directory if it does not currently exist |
| Replace existing files | If selected, duplicate files in the destination directory will be overwritten |
| Remove source files | If selected, removes the source files after copy (a move procedure) |
| Copy previous results to args | If selected, will use previous step results as your sources and destinations |
| File/folder source | The file or directory to copy from; click Browse and select Hadoop to enter your Hadoop cluster connection details |
| File/folder destination | The file or directory to copy to; click Browse and select Hadoop to enter your Hadoop cluster connection details |
| Wildcard (RegExp) | Defines the files that are copied in regular expression terms (instead of static file names), for instance: *.*.txt would be any file with a .txt extension |
| Files/folders | A list of selected sources and destinations |

Result files name

| Option | Definition |
|--------------------------------|---|
| Add files to result files name | Any files that are copied will appear as a result from this step; shows a list of files that were copied in this step |

Hadoop Job Executor

This job entry executes Hadoop jobs on a Hadoop node. There are two option modes: **Simple** (the default condition), in which you only pass a premade Java JAR to control the job; and **Advanced**, in which you are able to specify static main method parameters. Most of the options explained below are only available in Advanced mode. The **User Defined** tab in Advanced mode is for Hadoop option name/value pairs that are not defined in the **Job Setup** and **Cluster** tabs.

General

| Option | Definition |
|------------------------|---|
| Name | The name of this Hadoop Job Executor step instance. |
| Hadoop Job Name | The name of the Hadoop job you are executing. |
| Jar | The Java JAR that contains your Hadoop mapper and reducer job instructions in a static main method. |
| Command line arguments | Any command line arguments that must be passed to the static main method in the specified JAR. |

Job Setup

| Option | Definition |
|--------------------|--|
| Output Key Class | The Apache Hadoop class name that represents the output key's data type. |
| Output Value Class | The Apache Hadoop class name that represents the output value's data type. |
| Mapper Class | The Java class that will perform the map operation. Pentaho's default mapper class should be sufficient for most needs. Only change this value if you are supplying your own Java class to handle mapping. |
| Combiner Class | The Java class that will perform the combine operation. Pentaho's default combiner class should be sufficient for most needs. Only change this value if you are supplying your own Java class to handle combining. |
| Reducer Class | The Java class that will perform the reduce operation. Pentaho's default reducer class should be sufficient for most needs. Only change this value if you are supplying your own Java class to handle reducing. If you do not define a reducer class , then no reduce operation will be performed and the mapper or combiner output will be returned. |
| Input Path | The path to your input file on the Hadoop cluster. |

| Option | Definition |
|---------------|---|
| Output Path | The path to your output file on the Hadoop cluster. |
| Input Format | The Apache Hadoop class name that represents the input file's data type. |
| Output Format | The Apache Hadoop class name that represents the output file's data type. |

Cluster

| Option | Definition |
|-------------------------|--|
| Working Directory | The temporary job work directory on your Hadoop cluster. |
| HDFS Hostname | Hostname for your Hadoop cluster. |
| HDFS Port | Port number for your Hadoop cluster. |
| Job Tracker Hostname | If you have a separate job tracker node, type in the hostname here. Otherwise use the HDFS hostname. |
| Job Tracker Port | Job tracker port number; this cannot be the same as the HDFS port number. |
| Number of Mapper Tasks | The number of mapper tasks you want to assign to this job. The size of the inputs should determine the number of mapper tasks. Typically there should be between 10-100 maps per node, though you can specify a higher number for mapper tasks that are not CPU-intensive. |
| Number of Reducer Tasks | The number of reducer tasks you want to assign to this job. Lower numbers mean that the reduce operations can launch immediately and start transferring map outputs as the maps finish. The higher the number, the quicker the nodes will finish their first round of reduces and launch a second round. Increasing the number of reduce operations increases the Hadoop framework overhead, but improves load balancing. If this is set to 0 , then no reduce operation is performed, and the output of the mapper will be returned; also, combiner operations will also not be performed. |
| Enable Blocking | Forces the job to wait until each step completes before continuing to the next step. This is the only way for PDI to be aware of a Hadoop job's status. If left unchecked, the Hadoop job is blindly executed, and PDI moves on to the next step. Error handling/routing will not work unless this option is checked. |
| Logging Interval | Number of seconds between log messages. |

Hadoop Transformation Job Executor

This job entry executes transformations that require Hadoop data sources. This is frequently used to execute transformations that act as mappers and reducers in lieu of a traditional Hadoop Java class. The **User Defined** tab is for Hadoop option name/value pairs that are not defined in the **Job Setup** and **Cluster** tabs.

General

| Option | Definition |
|-----------------|--|
| Name | The name of this Hadoop Job Executor step instance |
| Hadoop Job Name | The name of the Hadoop job you are executing |

Map/Reduce

| Option | Definition |
|---------------------------|--|
| Mapper Transformation | The KTR that will perform the mapping functions for this job. |
| Mapper Input Step Name | The name of the step that receives mapping data from Hadoop. This must be an injector step. |
| Mapper Output Step Name | The name of the step that passes mapping output back to Hadoop. This must be a dummy step. |
| Combiner Transformation | The KTR that will perform the combiner functions for this job. |
| Combiner Input Step Name | The name of the step that receives combiner data from Hadoop. This must be an injector step. |
| Combiner Output Step Name | The name of the step that passes combiner output back to Hadoop. This must be a dummy step. |
| Reducer Transformation | The KTR that will perform the reducer functions for this job. |
| Reducer Input Step Name | The name of the step that receives reducing data from Hadoop. This must be an injector step. |
| Reducer Output Step Name | The name of the step that passes reducing output back to Hadoop. This must be a dummy step. |

Job Setup

| Option | Definition |
|--------------------|--|
| Output Key Class | The Apache Hadoop class name that represents the output key's data type. |
| Output Value Class | The Apache Hadoop class name that represents the output value's data type. |
| Input Path | The path to your input file on the Hadoop cluster. |
| Output Path | The path to your output file on the Hadoop cluster. |
| Input Format | The Apache Hadoop class name that represents the input file's data type. |
| Output Format | The Apache Hadoop class name that represents the output file's data type. |

Cluster

| Option | Definition |
|-------------------------|--|
| Working Directory | The temporary job work directory on your Hadoop cluster. |
| HDFS Hostname | Hostname for your Hadoop cluster. |
| HDFS Port | Port number for your Hadoop cluster. |
| Job Tracker Hostname | If you have a separate job tracker node, type in the hostname here. Otherwise use the HDFS hostname. |
| Job Tracker Port | Job tracker port number; this cannot be the same as the HDFS port number. |
| Number of Mapper Tasks | The number of mapper tasks you want to assign to this job. The size of the inputs should determine the number of mapper tasks. Typically there should be between 10-100 maps per node, though you can specify a higher number for mapper tasks that are not CPU-intensive. |
| Number of Reducer Tasks | The number of reducer tasks you want to assign to this job. Lower numbers mean that the reduce operations can launch immediately and start transferring map outputs as the maps finish. The higher the number, the quicker the nodes will finish their first round of reduces and launch a second round. Increasing the number of reduce operations increases the Hadoop framework overhead, but improves load balancing. If this is set to 0 , then no reduce operation is performed, and the output of the mapper will be returned; also, combiner operations will also not be performed. |
| Enable Blocking | Forces the job to wait until each step completes before continuing to the next step. This is the only way for PDI to be aware of a Hadoop job's status. If left unchecked, the Hadoop job is blindly executed, and PDI moves on to the next step. Error handling/routing will not work unless this option is checked. |
| Logging Interval | Number of seconds between log messages. |

Amazon EMR Job Executor

This job entry executes Hadoop jobs on an Amazon Elastic MapReduce (EMR) account. In order to use this step, you must have an Amazon Web Services (AWS) account configured for EMR, and a premade Java JAR to control the remote job.

| Option | Definition |
|-------------------|--|
| Name | The name of this Amazon EMR Job Executor step instance. |
| EMR Job Flow Name | The name of the Amazon EMR job flow (series of steps) you are executing. |

| Option | Definition |
|------------------------|---|
| AWS Access Key | Your Amazon Web Services access key. |
| AWS Secret Key | Your Amazon Web Services secret key. |
| S3 Staging Directory | The Amazon Simple Storage Service (S3) address of the working directory for this Hadoop job. This directory will contain the MapReduce JAR, and log files will be placed here as they are created. |
| MapReduce JAR | The Java JAR that contains your Hadoop mapper and reducer classes. The job must be configured and submitted using a static main method in any class in the JAR. |
| Command line arguments | Any command line arguments that must be passed to the static main method in the specified JAR. |
| Number of Instances | The number of Amazon Elastic Compute Cloud (EC2) instances you want to assign to this job. |
| Master Instance Type | The Amazon EC2 instance type that will act as the Hadoop "master" in the cluster, which handles map/reduce task distribution. |
| Slave Instance Type | The Amazon EC2 instance type that will act as one or more Hadoop "slaves" in the cluster. Slaves are assigned tasks from the master. This is only valid if the number of instances is greater than 1. |
| Enable Blocking | Forces the job to wait until each step completes before continuing to the next step. This is the only way for PDI to be aware of a Hadoop job's status. If left unchecked, the Hadoop job is blindly executed, and PDI moves on to the next step. Error handling/routing will not work unless this option is checked. |
| Logging Interval | Number of seconds between log messages. |

Troubleshooting

This section contains information about changing the Kettle Home directory. More troubleshooting tips will be added to this document in the future.

Changing the Pentaho Data Integration Home Directory Location (.kettle folder)

The default Pentaho Data Integration (PDI) HOME directory is the user's home directory (for example, in Windows C:\Documents and Settings\{user}\.kettle or for all other *nix based operating systems (\$HOME/.kettle)). The directory may change depending on the user who is logged on. As a result, the configuration files that control the behavior of PDI jobs and transformations are different from user to user. This also applies when running PDI from the Pentaho BI Platform.

When you set the KETTLE_HOME variable, the PDI jobs and transformations can be run without being affected by the user who is logged on. KETTLE_HOME is used to change the location of the files normally in [user home].kettle.



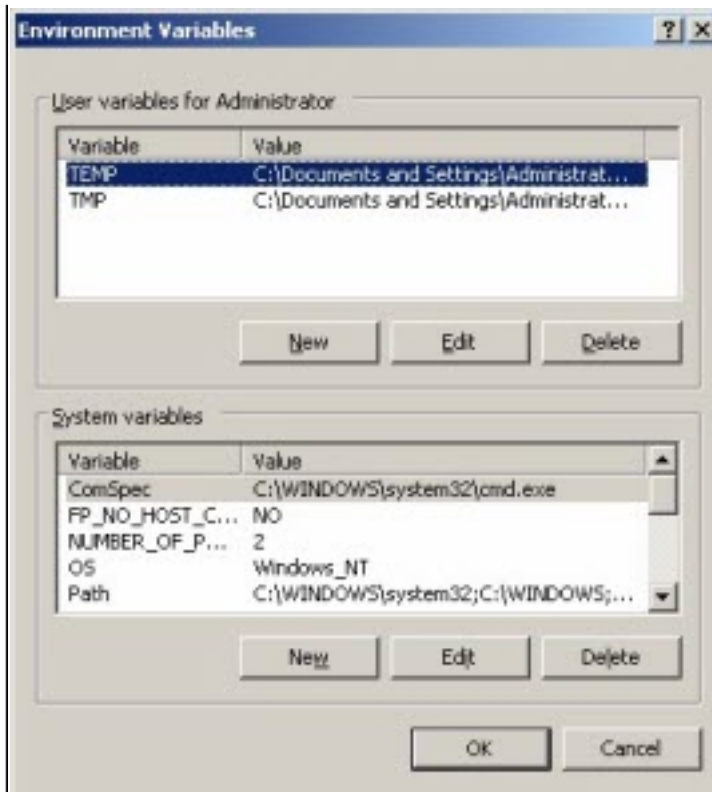
Note: The PDI home directory is independent of the PDI application directory.

Below is a short description of each item in the HOME directory:

| Item | Description |
|--------------------------|---|
| kettle.properties | Default properties file for variables |
| shared.xml | Default shared objects file |
| db.cache | The database cache for metadata |
| repositories.xml | The local repositories file |
| .spoonrc | User interface settings, last opened transformation/job |
| .languageChoice | User language (delete to revert language) |

To change set the KETTLE_HOME variable...

| Step | Description |
|---------------|---|
| Step 1 | Set the KETTLE_HOME variable according to your needs. This can be performed system wide by the operating system or just before the start of PDI using a shell script or batch (for example, use the SET command). The KETTLE_HOME variable can be set system wide on Windows systems using the environment variables settings (see below): |



Step 2 Point the KETTLE_HOME to the directory that contains the .kettle directory. The .kettle gets appended by PDI. For example, when you have stored the common files in C:\Pentaho\Kettle\common\.kettle you need to set the KETTLE_HOME variable to C:\Pentaho\Kettle\common).


The method above can also be used for configuration management and deployment to switch between the test, development, and production environments when other variables like the database server of the connection is defined in kettle.properties.

For testing purposes set a variable in the kettle.properties file of your defined .kettle home directory. Set the KETTLE_HOME directory accordingly by using the operating system SET command. Start Spoon and go to **Edit -> Show Environment Variables**. You should see the variables defined in kettle.properties.

Changing the Kettle Home Directory within the Pentaho BI Platform

You can set the KETTLE_HOME directory in the BI Server:

1. When started as a service, edit the registry: HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software Foundation\Procrun 2.0\pentahobiserver\Parameters\Java

 **Note:** On 64-bit systems, the Apache Software Foundation is under **Wow6432Node**.

2. Add a new line (**not a space!**) to the **Options** associated with the KETTLE_HOME variable, for example, -Dcatalina.base=C:\Pentaho\3.0.1-GA\Installed\server\biserver-ee\tomcat

```
[...]
-XX:MaxPermSize=256m
-DKETTLE_HOME=C:\Pentaho\Kettle\KETTLE_HOME
```

3. Reboot the server.
4. When you start the BI Server from the command line, you must edit the ...server\biserver-ee\start-pentaho.bat (see below):

```
[...]
set CATALINA_HOME=%PENTAHO_PATH%tomcat
set CATALINA_OPTS=-Xms256m -Xmx768m -XX:MaxPermSize=256m -
Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -
DKETTLE_HOME=C:\Pentaho\Kettle\KETTLE_HOME
call startup
endlocal
:quit
```

FAQ: Pentaho Data Integration 4.1

Pentaho customers have frequently asked the following questions about Pentaho Data Integration 4.1:

Are DI Server schedules persistent? That is, if the machine or just the DI Server is restarted, then will it pick up the schedule for any jobs that need to be run? For example, I have 10 jobs set to run daily at midnight. If the DI Server is restarted at 2PM, will it pick up the scheduled jobs for midnight?

Yes; schedules are stored in the DI Server quartz database.

Can I auto-register the DI Server server in Pentaho Enterprise Console?

Yes. You can add a SQL statement like the one shown below to the `/pentaho/server/enterprise-console/resource/config/hsqldb/pentaho-mgmt-svcs.script` script.

```
INSERT INTO PAC_PROPERTIES VALUES('PDI','pdi.carte.username','joe')
INSERT INTO PAC_PROPERTIES VALUES('PDI','pdi.carte.password','password')
INSERT INTO PAC_PROPERTIES VALUES('PDI','pdi.carte.service','http://localhost:9080/pentaho-di/kettle/')
```

Is Enterprise Console connected to Carte, or to the DI Server? It asks to register for Carte, but with PDI 4.1 I would prefer to connect it to the DI Server instead so that my jobs are monitored in Enterprise Console.

When you register a remote execution server in Enterprise Console, it can be a Carte server or a Data Integration Server. The registration dialog says "Carte," but it will accept a DI Server server registration as well. Only one remote execution server can be registered at a time through Enterprise Console.

When jobs are started in Spoon via the Schedule function in the Action menu, are they stored on the DI Server? Will they be visible through Enterprise Console?

Yes, they go by default to your DI Server as defined by the enterprise repository connection. Executions are visible through Enterprise Console as long as the DI Server server that executes the schedule is the same DI Server server that is registered in Enterprise Console.

I do not want to register jobs in Enterprise Console, but Enterprise Console should show all running and idle jobs in the repository. Is there a way to configure this?

Enterprise Console will show step statistics and Carte logging for all jobs and transformations executed on the registered remote execution server. Registering a transformation or a job in Enterprise Console allows you to see additional logged information, like execution history and performance trends. You cannot currently see the entire list of idle or available jobs without using the registration dialogs.

Enterprise Console is our main job management and monitoring tool. Can I schedule jobs directly from Enterprise Console?

Only by wrapping the transformation in an action sequence, and using the BI Server's scheduling feature. DI Server scheduling is only available through Spoon.

Does the enterprise repository always use the local H2 database, or can the database be remote and of a different type, like SQL Server, etc.? My old Kettle database repository was able to reside on any DB type and on a remote machine.

The enterprise repository persistence today is delegated to the JackRabbit CMS. You can read about the options for persistence here: <http://wiki.apache.org/jackrabbit/PersistenceManagerFAQ#ObjectPersistenceManager>.

Any changes you may make to the enterprise repository implementation or configuration are not supported by Pentaho. This is a "black box" offering that does not allow for extra configuration support.