The Pentaho Big Data Guide

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

Trademarks

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

This document contains instructions for installing, configuring, and using the Big Data functionality in Pentaho Data Integration 4.3. This encompasses running PDI on a Hadoop node, executing Hadoop jobs via PDI, and accessing Hadoop, MongoDB, Cassandra, and HBase as PDI data sources.

Note: This is a tiny subset of the total Pentaho Data Integration documentation. It is designed to help evaluators, developers, and new customers who are only interested in PDI for its Big Data functions. If you require more general PDI documentation, consult the complete PDI documentation set: *Pentaho Data Integration User Guide*, *Pentaho Data Integration Administrator's Guide*, and *Installing Pentaho Data Integration*. For a beginner's walkthrough of PDI 4.3, refer instead to *Getting Started With Pentaho Data Integration*.

Supported Big Data Technologies

Pentaho supports the following **Hadoop** distributions:

- Apache Hadoop 0.20.2 and 0.20.203.0
- Cloudera CDH3u2 and CDH3u3
- MapR 1.1.3 and 1.2.0

And the following **Cassandra** distributions:

- Apache 1.0.9
- DataStax 1.0.8

And the following other Big Data technologies:

- Hive 0.7.1
- MongoDB 2.0.4
- HBase 0.90.5

Installing PDI For Hadoop

Below are instructions for installing PDI on user workstations and, if necessary, patching it to support specific Hadoop distributions. Installation and configuration of the Data Integration Server are not covered in this document; there is no special Hadoop functionality in the DI Server.

Note: By default, Business Analytics is configured to work with Apache Hadoop. If you are using Cloudera or MapR, there are special instructions in this section to switch over to them instead.

Obtaining the Installation Materials

Note: All Big Data functionality is now included natively in all Pentaho servers and client tools. No special Big Data or Hadoop packages are required. The information below is for standard PDI Enterprise Edition packages.

Consult the Welcome Kit email that was sent to you after completing the sales process. This email contains user credentials for the Pentaho Customer Support Portal, where you can download individual archive packages for the Data Integration desktop client tools needed to design jobs and transformations for Big Data processes. Here are the packages you need for each platform and distribution:

- **Data Integration client tool Windows package:** `pdi-ee-client-4.3.0-GA.zip`
- **Data Integration client tool Linux/Solaris/OS X package:** `pdi-ee-client-4.3.0-GA.tar.gz`

Note: You may wish to install the Data Integration Server as well. The DI Server installation and configuration process is covered in the *Pentaho Data Integration Installation Guide*.

Workstation Archive Package Deployment

Follow the below instructions to install the Data Integration client tools on your workstations.

Note: The example commands in this and other sections are specific to Linux. You will have to adjust or ignore them on other operating systems.

1. Create a **/pentaho/design-tools/** directory in an appropriate place in your hierarchy.

   This directory should be accessible to all of the user accounts on this system that will run PDI.

   Note: If you are using the graphical installer, it will create this directory structure for you, so you can skip this step.

   ```
   mkdir -p /usr/local/pentaho/design-tools/
   ```
2. Unpack the **pdi-ee-client-4.3.0-GA** archive to `/pentaho/design-tools/`.

   ```
   tar zxvf pdi-ee-client-4.3.0-GA.tar.gz -C /usr/local/pentaho/design-tools/
   ```
3. Ensure that the **hadoop-core** JAR in your Hadoop node is the same version as the one Pentaho ships with PDI (in `/data-integration/libext/bigdata/`).

   If these JARs are different versions, strange and unusual problems can occur.
4. Navigate to the `/pentaho/design-tools/license-installer/` directory.
5. Run the **install_license.sh** script with the sole parameter of the location and name of your license file.

   ```
   ./install_license.sh install /home/rwilco/downloads/Pentaho\ PDI\ Enterprise\
    Edition.lic
   ```

The Data Integration client tools are now installed.

Configuring For MapR

Follow the below instructions to modify your PDI client tool and Report Designer to connect to MapR Hadoop.

Note: Because MapR requires certain JARs that are mutually exclusive with other Hadoop distributions, you will not be able to connect to any Hadoop distribution other than MapR unless you install a second instance of the PDI client tool.

1. Delete the `/pentaho/design-tools/data-integration/libext/bigdata/hadoop-0.20.2-core.jar` file.
2. Copy the `/hadoop/hadoop-0.20.2/lib/hadoop-0.20.2-dev-core.jar` file from your MapR directory to the `/pentaho/design-tools/data-integration/libext/bigdata/` directory.
3. Copy the `/hadoop/hadoop-0.20.2/lib/maprfs-0.1.jar` file from your MapR directory to the `/pentaho/design-tools/data-integration/libext/bigdata/` directory.
4. Edit the `/pentaho/design-tools/data-integration/launcher/launcher.properties` file, and add the following entry to the end of the **libraries** path:

   ```
   :../../../../../../../../../../../../../opt/mapr/hadoop/hadoop-0.20.2/conf
   ```
5. At the bottom of the launcher.properties file, add these three lines:

   ```
   #-Djava.security.krb5.realm=OX.AC.UK -
   Djava.security.krb5.kdc=kdc0.ox.ac.uk:kdc1.ox.ac.uk
   system-property.java.security.krb5.realm=OX.AC.UK
   system-property.java.security.krb5.kdc=kdc0.ox.ac.uk:kdc1.ox.ac.uk
   ```
6. If you are running PDI on OS X, edit the `/pentaho/design-tools/data-integration/Data Integration 64-bit App/Contents/info.plist` file and replace the last **<string>** element with the one shown below:

   ```
   <string>-Xmx256m -Xms256m -XX:MaxPermSize=128m -Djava.library.path=/opt/mapr/hadoop/
   hadoop-0.20.2/lib/native/Mac_OS_X-x86_64-64</string>
   ```
7. If you are on 64-bit Linux, edit the following scripts in the `/pentaho/design-tools/data-integration/` directory and add the below lines to each of them, immediately before the **OPT=** path:

   - carte.sh
   - kitchen.sh
   - pan.sh
   - spoon.sh

   ```
   # Add MapR Linux x64 native libraries to the path
   # For 32 bit libraries replace Linux-amd64-64 with Linux-i386-32
   LIBPATH=$LIBPATH:/opt/mapr/hadoop/hadoop-0.20.2/lib/native/Linux-amd64-64

   export LIBPATH
   ```
8. Delete the `/pentaho/design-tools/report-designer/lib/jdbc/hadoop-0.20.2-core.jar` file.
9. Copy the `/hadoop/hadoop-0.20.2/lib/hadoop-0.20.2-dev-core.jar` from your MapR directory to the `/pentaho/design-tools/report-designer/lib/` directory.
10. Copy the `/hadoop/hadoop-0.20.2/lib/maprfs-0.1.jar` from your MapR directory to the `/pentaho/design-tools/report-designer/lib/` directory.
11. If you are on 64-bit Linux, add the following text to the last line of the `/pentaho/design-tools/report-designer/report-designer.sh` script:

    ```
    -Djava.library.path=/opt/mapr/hadoop/hadoop-0.20.2/lib/native/Linux-amd64-64
    ```
12. If you are on OS X, edit the `/Pentaho Report Designer.app/Contents/Info.plist` file and add the following text to the **VMOptions** element:

    ```
    -Djava.library.path=/opt/mapr/hadoop/hadoop-0.20.2/lib/native/Mac_OS_X-x86_64-64
    ```

The Pentaho Data Integration client tool (Spoon) and Report Designer are now configured to connect to MapR instead of Apache Hadoop.

Configuring For Cloudera CDH3

You must have a Cloudera CDH Update 3 (CDH3 or CDHu3) distribution in order to proceed. It may be possible to use this procedure generically to connect to Hadoop distributions that Pentaho does not officially support.

Follow the below instructions to modify your PDI client tool to connect to a Cloudera Hadoop node.

Note: Because Cloudera requires certain JARs that are mutually exclusive with other Hadoop distributions, you will not be able to connect to any Hadoop distribution other than Cloudera unless you install a second instance of the PDI client tool.

1. Delete the `/pentaho/design-tools/data-integration/libext/bigdata/hadoop-0.20.2-core.jar` file.
2. Copy the `/hadoop/hadoop-0.20.2/lib/hadoop-core-0.20.2-cdh3u3.jar` file from your Cloudera directory to the `/pentaho/design-tools/data-integration/libext/bigdata/` directory.
3. You must download the Apache Commons Configuration JAR (commons-configuration-1.7.jar) from *http://commons.apache.org/configuration/download_configuration.cgi* and copy it to the `/pentaho/design-tools/data-integration/libext/commons/` directory.
4. Copy the `/lib/guava-r09-jarjar.jar` file from your Cloudera directory to the `/pentaho/design-tools/data-integration/libext/bigdata/` directory.
5. Delete the `/pentaho/design-tools/report-designer/lib/bigdata/hadoop-0.20.2-core.jar` file.
6. Copy the `/hadoop/hadoop-0.20.2/lib/hadoop-core-0.20.2-cdh3u3.jar` file from your Cloudera directory to the `/pentaho/design-tools/report-designer/lib/bigdata/` directory.
7. You must download (or copy from PDI) the Apache Commons Configuration JAR (commons-configuration-1.7.jar) from *http://commons.apache.org/configuration/download_configuration.cgi* and copy it to the `/pentaho/design-tools/report-designer/lib/commons/` directory.
8. Copy the `/lib/guava-r09-jarjar.jar` file from your Cloudera directory to the `/pentaho/design-tools/report-designer/lib/bigdata/` directory.
9. Delete the `/WEB-INF/lib/hadoop-0.20.2-core.jar` file inside of the deployed **pentaho.war**.

   For graphical installer and archive deployments, this is inside of Tomcat. The path for the exploded WAR is: `/pentaho/server/biserver-ee/tomcat/webapps/pentaho/`
10. Copy the `/hadoop/hadoop-0.20.2/lib/hadoop-core-0.20.2-cdh3u3.jar` file from your Cloudera directory to the `/WEB-INF/lib/` directory inside of the **pentaho.war**.
11. You must download (or copy from PDI) the Apache Commons Configuration JAR (commons-configuration-1.7.jar) from *http://commons.apache.org/configuration/download_configuration.cgi* and copy it to the `/WEB-INF/lib/` directory inside of the **pentaho.war**.

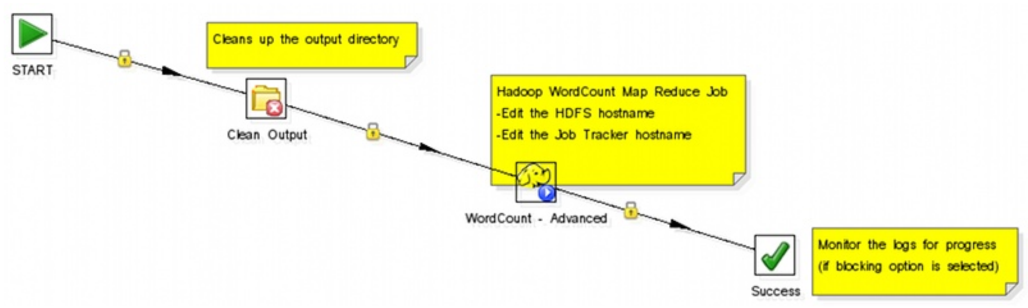
The Pentaho Data Integration client tool (Spoon), Report Designer, and the BA Server are now configured to work with the Cloudera CDH3 Hadoop distribution.

Cleanup

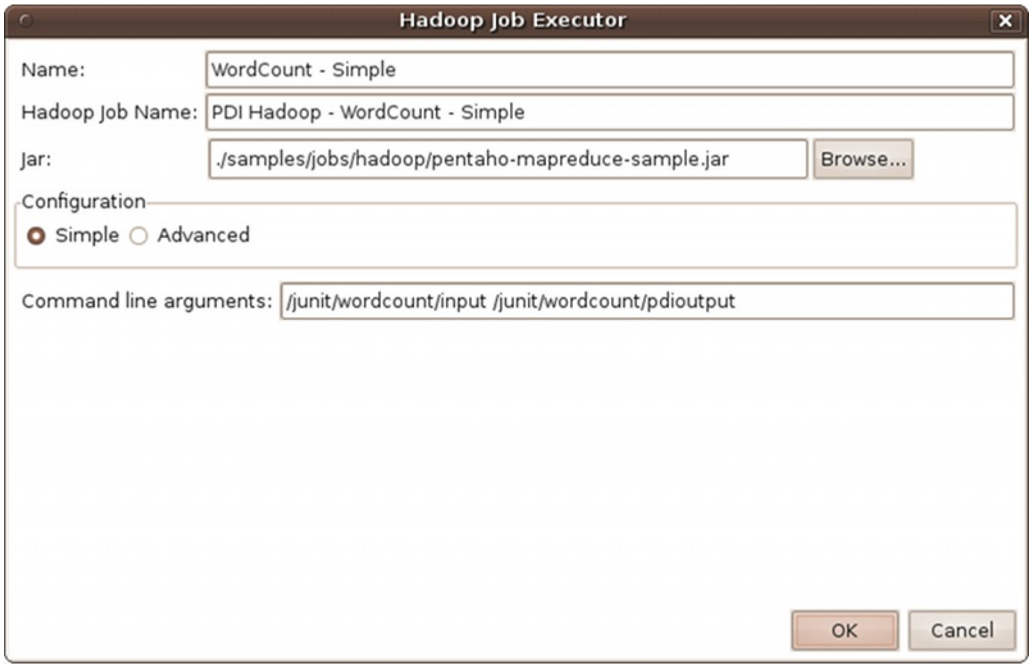
You can now delete the archive packages you downloaded.

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**:

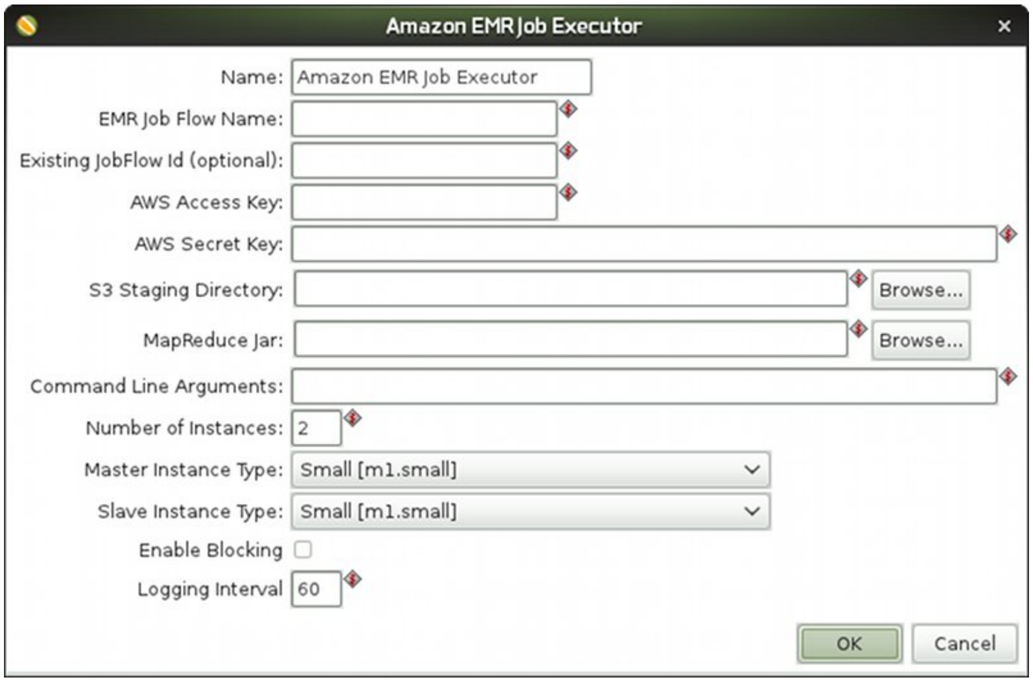

START
Cleans up the output directory
Clean Output
Hadoop WordCount Map Reduce Job
-Edit the HDFS hostname
-Edit the Job Tracker hostname
WordCount - Advanced
Success
Monitor the logs for progress
(if blocking option is selected)

This step requires a custom mapper/reducer Java class:



Name: WordCount - Simple
Hadoop Job Name: PDI Hadoop - WordCount - Simple
Jar: ./samples/jobs/hadoop/pentaho-mapreduce-sample.jar
Configuration
[✓ Simple] Advanced
Command line arguments: /junit/wordcount/input /junit/wordcount/pdioutput

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:

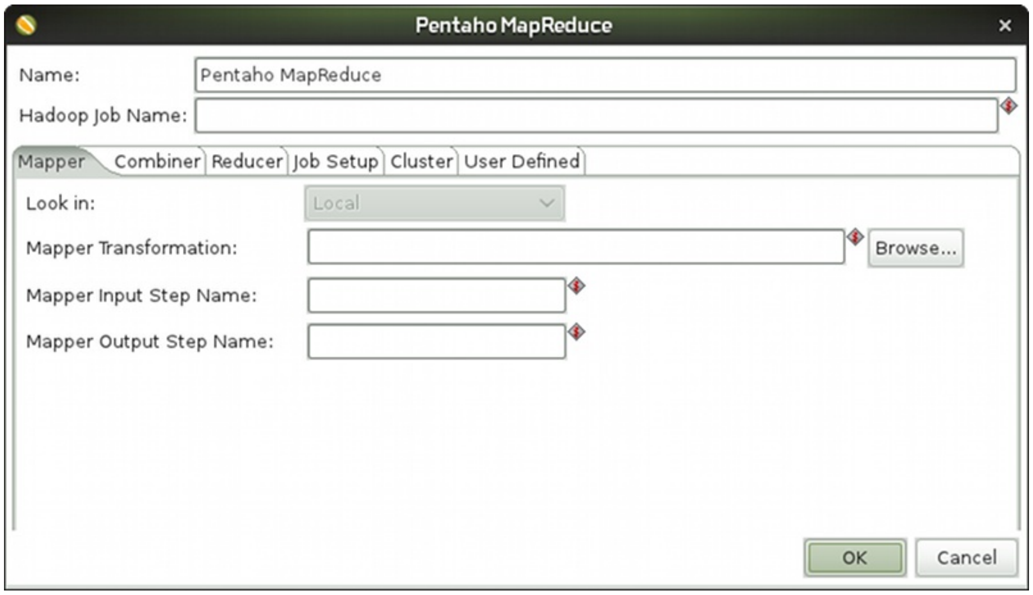


Amazon EMR Job Executor

Name: Amazon EMR Job Executor
EMR Job Flow Name:
Existing JobFlow Id (optional):
AWS Access Key:
AWS Secret Key:
S3 Staging Directory: Browse...
MapReduce Jar: Browse...
Command Line Arguments:
Number of Instances: 2
Master Instance Type: Small [m1.small]
Slave Instance Type: Small [m1.small]
Enable Blocking ☐
Logging Interval 60

OK Cancel

You can also execute a PDI job that includes Hadoop-oriented transformations through the **Pentaho MapReduce**. 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:



Pentaho MapReduce

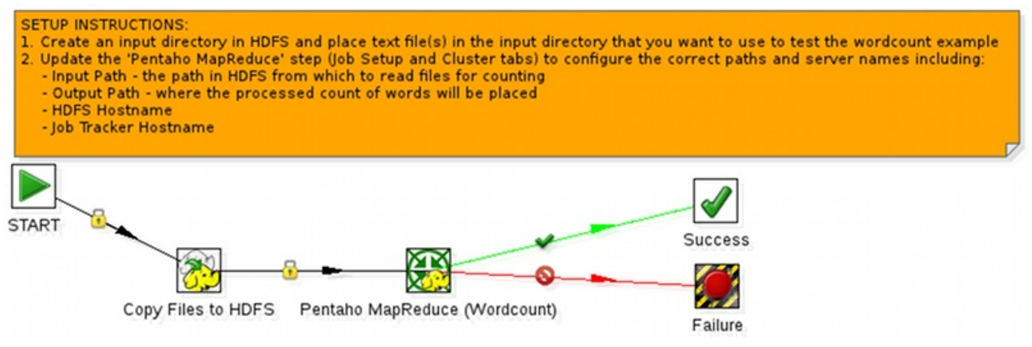
Name: Pentaho MapReduce
Hadoop Job Name:

Mapper | Combiner | Reducer | Job Setup | Cluster | User Defined

Look in: Local
Mapper Transformation: Browse...
Mapper Input Step Name:
Mapper Output Step Name:

OK Cancel

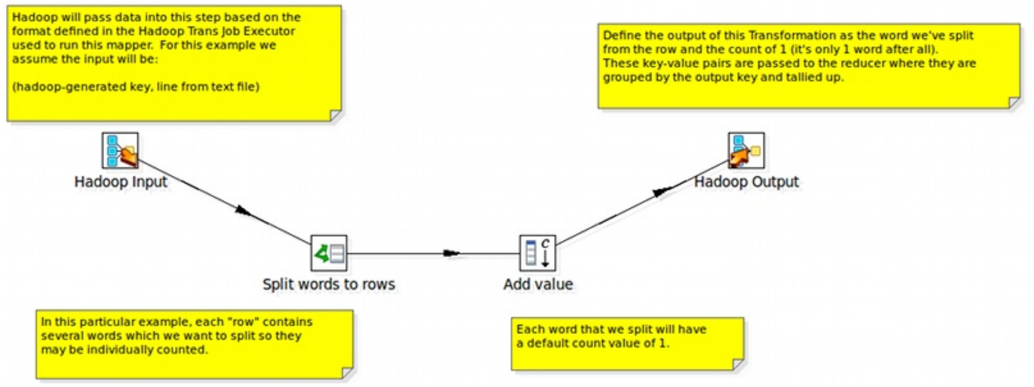
The workflow for the transformation job executor looks something like this:



SETUP INSTRUCTIONS:
1. Create an input directory in HDFS and place text file(s) in the input directory that you want to use to test the wordcount example
2. Update the 'Pentaho MapReduce' step (Job Setup and Cluster tabs) to configure the correct paths and server names including:
- Input Path - the path in HDFS from which to read files for counting
- Output Path - where the processed count of words will be placed
- HDFS Hostname
- Job Tracker Hostname

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 pass data into this step based on the format defined in the Hadoop Trans Job Executor used to run this mapper. For this example we assume the input will be:

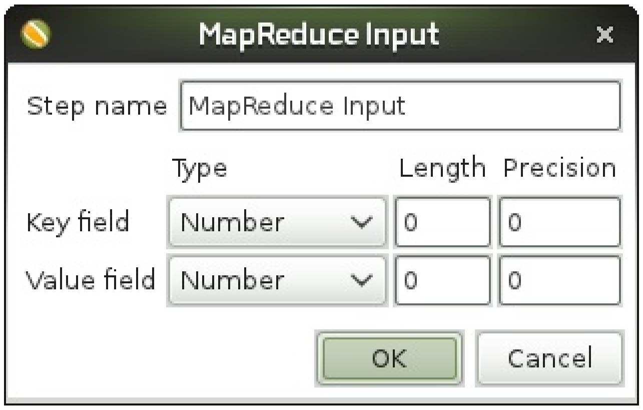
(hadoop-generated key, line from text file)

Define the output of this Transformation as the word we've split from the row and the count of 1 (it's only 1 word after all). These key-value pairs are passed to the reducer where they are grouped by the output key and tallied up.

In this particular example, each "row" contains several words which we want to split so they may be individually counted.

Each word that we split will have a default count value of 1.

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

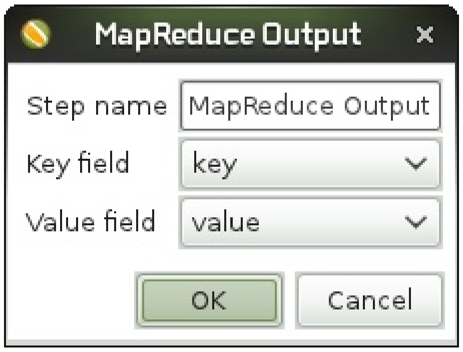


MapReduce Input

Step name: MapReduce Input

	Type	Length	Precision
Key field	Number	0	0
Value field	Number	0	0

...and a **MapReduce Output** step that passes the output back to Hadoop.



MapReduce Output

Step name: MapReduce Output

Key field: key

Value field: value

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

PDI Transformation Step Reference

The transformation steps explained below pertain to Hadoop functions in Pentaho Data Integration.

Calculator Input

Configure Calculator Input

This calculator step provides you with predefined functions that can be executed on input field values.

Besides the arguments (Field A, Field B and Field C) you must also specify the return type of the function. You can also choose to remove the field from the result (output) after all values are calculated.

Function	Description	Required fields
Set field to constant A	Create a field with a constant value.	A
A + B	A plus B.	A and B
A - B	A minus B.	A and B
A * B	A multiplied by B.	A and B
A / B	A divided by B.	A and B
A * A	The square of A.	A
SQRT(A)	The square root of A.	A
100 * A / B	Percentage of A in B.	A and B
A - (A * B / 100)	Subtract B% of A.	A and B
A + (A * B / 100)	Add B% to A.	A and B
A + B *C	Add A and B times C.	A, B and C
SQRT(A*A + B*B)	Calculate ?(A2+B2).	A and B
ROUND(A)	Round A to the nearest integer.	A
ROUND(A, B)	Round A to B decimal positions.	A and B
NVL(A, B)	If A is not NULL, return A, else B. Note that sometimes your variable won't be null but an empty string.	A and B
Date A + B days	Add B days to Date field A.	A and B
Year of date A	Calculate the year of date A.	A
Month of date A	Calculate number the month of date A.	A
Day of year of date	A Calculate the day of year (1-365).	A
Day of month of date A	Calculate the day of month (1-31).	A
Day of week of date A	Calculate the day of week (1-7).	A
Week of year of date A	Calculate the week of year (1-54).	A
ISO8601 Week of year of date A	Calculate the week of the year ISO8601 style (1-53).	A
ISO8601 Year of date A	Calculate the year ISO8601 style.	A
Byte to hex encode of string A	Encode bytes in a string to a hexadecimal representation.	A
Hex encode of string A	Encode a string in its own hexadecimal representation.	A
Char to hex encode of string A	Encode characters in a string to a hexadecimal representation.	A
Hex decode of string A	Decode a string from its hexadecimal representation (add a leading 0 when A is of odd length).	A
Checksum of a file A using CRC-32	Calculate the checksum of a file using CRC-32.	A
Checksum of a file A using Adler-32	Calculate the checksum of a file using Adler-32.	A
Checksum of a file A using MD5	Calculate the checksum of a file using MD5.	A

Function	Description	Required fields
Checksum of a file A using SHA-1	Calculate the checksum of a file using SHA-1.	A
Levenshtein Distance (Source A and Target B)	Calculates the Levenshtein Distance.	A and B
Metaphone of A (Phonetics)	Calculates the metaphone of A.	A
Double metaphone of A	Calculates the double metaphone of A.	A
Absolute value ABS(A)	Calculates the Absolute value of A.	A
Remove time from a date A	Removes time value of A.	A
Date A - Date B (in days)	Calculates difference, in days, between A date field and B date field.	A and B
A + B + C	A plus B plus C.	A, B, and C
First letter of each word of a string A in capital	Transforms the first letter of each word within a string.	A
UpperCase of a string A	Transforms a string to uppercase.	A
LowerCase of a string A	Transforms a string to lowercase.	A
Mask XML content from string A	Escape XML content; replace characters with &values.	A
Protect (CDATA) XML content from string A	Indicates an XML string is general character data, rather than non-character data or character data with a more specific, limited structure. The given string will be enclosed into <![CDATA[String]]>.	A
Remove CR from a string A	Removes carriage returns from a string.	A
Remove LF from a string A	Removes linefeeds from a string.	A
Remove CRLF from a string A	Removes carriage returns/linefeeds from a string.	A
Remove TAB from a string A	Removes tab characters from a string.	A
Return only digits from string A	Outputs only Outputs only digits (0-9) from a string from a string.	A
Remove digits from string A	Removes all digits (0-9) from a string.	A
Return the length of a string A	Returns the length of the string.	A
Load file content in binary	Loads the content of the given file (in field A) to a binary data type (e.g. pictures).	A
Add time B to date A	Add the time to a date, returns date and time as one value.	A and B
Quarter of date A	Returns the quarter (1 to 4) of the date.	A
variable substitution in string A	Substitute variables within a string.	A
Unescape XML content	Unescape XML content from the string.	A
Escape HTML content	Escape HTML within the string.	A
Unescape HTML content	Unescape HTML within the string.	A
Escape SQL content	Escapes the characters in a String to be suitable to pass to an SQL query.	A
Date A - Date B (working days)	Calculates the difference between Date field A and Date field B (only working days Mon-Fri).	A and B
Date A + B Months	Add B months to Date field A.	A
Check if an XML file A is well formed	Validates XML file input.	A
Check if an XML string A is well formed	Validates XML string input.	A
Get encoding of file A	Guess the best encoding (UTF-8) for the given file.	A
Dameraulevenshtein distance between String A and String B	Calculates Dameraulevenshtein distance between strings.	A and B
NeedlemanWunsch distance between String A and String B	Calculates NeedlemanWunsch distance between strings.	A and B

Function	Description	Required fields
Jaro similitude between String A and String B	Returns the Jaro similarity coefficient between two strings.	A and B
JaroWinkler similitude between String A and String B	Returns the Jaro similarity coefficient between two strings.	A and B
SoundEx of String A	Encodes a string into a Soundex value.	A
RefinedSoundEx of String A	Retrieves the Refined Soundex code for a given string object	A
Date A + B Hours	Add B hours to Date field A	A and B
Date A + B Minutes	Add B minutes to Date field A	A and B
Date A - Date B (milliseconds)	Subtract B milliseconds from Date field A	A and B
Date A - Date B (seconds)	Subtract B seconds from Date field A	A and B
Date A - Date B (minutes)	Subtract B minutes from Date field A	A and B
Date A - Date B (hours)	Subtract B hours from Date field A	A and B

Cassandra Input

Configure Cassandra Input

Cassandra Input is an input step that allows data to be read from a Cassandra column family (table) as part of an ETL transformation.

Option	Definition
Step name	The name of this step as it appears in the transformation workspace.
Cassandra host	Connection host name input field.
Cassandra port	Connection host port number input field.
Username	Input field for target keyspace and/or family (table) authentication details.
Password	Input field for target keyspace and/or family (table) authentication details.
Keyspace	Input field for the keyspace (database) name.
Use query compression	If checked, tells the step whether or not to compress the text of the CQL query before sending it to the server.
Show schema	Opens a dialog that shows metadata for the column family named in the CQL SELECT query.

CQL SELECT Query

The large text box at the bottom of the dialog allows you to enter a CQL SELECT statement to be executed. Only a single SELECT query is accepted by the step.

```
SELECT [FIRST N] [REVERSED] <SELECT EXPR> FROM <COLUMN FAMILY> [USING
                 <CONSISTENCY>] [WHERE <CLAUSE>] [LIMIT N];
```

Important: Cassandra Input does not support the CQL range notation (e.g. name1..nameN) for specifying columns in a SELECT query.

Select queries may name columns explicitly (in a comma separated list) or use the * wildcard. If wildcard is used then only those columns (if any) defined in the metadata for the column family in question are returned. If columns are selected explicitly, then the name of each column must be enclosed in single quotation marks. Since Cassandra is a sparse column oriented database (like HBase), it is possible for rows to contain varying numbers of columns which might, or might not, be defined in the metadata for the column family. The Cassandra Input step can emit columns that are not defined in the metadata for the column family in question if they are explicitly named in the SELECT clause. Cassandra Input uses type information present in the metadata for a column family. This, at a minimum, includes a default type (column validator) for the column family. If there is explicit metadata for individual columns available, then this is used for type information, otherwise the default validator is used.

Option	Definition
LIMIT	If omitted, Cassandra assumes a default limit of 10,000 rows to be returned by the query. If the query is expected to return more than 10,000 rows an explicit LIMIT clause must be added to the query.
FIRST N	Returns the first N (as determined by the column sorting strategy used for the column family in question) column values from each row. If the column family in question is sparse then this may result in a different N (or less) column values appearing from one row to the next. Since PDI deals with a constant number of fields between steps in a transformation, Cassandra rows that do not contain particular columns are output as rows with null field values for non-existent columns. Cassandra's default for FIRST (if omitted from the query) is 10,000 columns - if a query is expected to return more than 10,000 columns then an explicit FIRST must be added to the query.
REVERSED	Option causes the sort order of the columns returned by Cassandra for each row to be reversed. This may affect which values result from a FIRST N option, but does not affect the order of the columns output by Cassandra Input.
WHERE clause	Clause provides for filtering the rows that appear in results. The clause can filter on a key name, or range of keys, and in the case of indexed columns, on column values. Key filters are specified using the KEY keyword, a relational operator, (one of =, >, >=, <, and <=), and a term value.

Cassandra Output

Configure Cassandra Output

Cassandra Output is an output step that allows data to be written to a Cassandra column family (table) as part of an ETL transformation.

Option	Definition
Step name	The name of this step as it appears in the transformation workspace.
Cassandra host	Connection host name input field.
Cassandra port	Connection host port number input field.
Username	Target keyspace and/or family (table) authentication details input field.
Password	Target keyspace and/or family (table) authentication details input field.
Keyspace	Input field for the keyspace (database) name.
Show schema	Opens a dialog that shows metadata for the specified column family.

Configure Column Family and Consistency Level

This tab contains connection details and basic query information, in particular, how to connect to Cassandra and execute a CQL (Cassandra Query Language) query to retrieve rows from a column family (table).

Important: Note that Cassandra Output does not check the types of incoming columns against matching columns in the Cassandra metadata. Incoming values are formatted into appropriate string values for use in a textual CQL INSERT statement according to PDI's field metadata. If resulting values can't be parsed by the Cassandra column validator for a particular column then an error will result.

Note: Cassandra Output converts PDI's dense row format into sparse data by ignoring incoming field values that are null.

Option	Definition
Column family (table)	Input field to specify which column family the incoming rows should be written to.
Get column family names button	Populates the drop-down box with names of all the column families that exist in the specified keyspace.
Consistency level	Input field enables an explicit write consistency to be specified. Valid values are: ZERO, ONE, ANY, QUORUM and ALL. The Cassandra default is ONE.
Create column family	If checked, allows the step to create the named column family if it does not already exist.
Truncate column family	If checked, specifies whether any existing data should be deleted from the named column family before inserting incoming rows.
Update column family metadata	If checked, updates the column family metadata with information on incoming fields not already present, when option is selected. If this option is not selected, then any unknown incoming fields are ignored unless the Insert fields not in column metadata option is enabled.
Insert fields not in column metadata	If checked, inserts the column family metadata in any incoming fields not present, with respect to the default column family validator. This option has no effect if Update column family metadata is selected.
Commit batch size	Allows you to specify how many rows to buffer before executing a BATCH INSERT CQL statement.
Use compression	Option will compress (gzip) the text of each BATCH INSERT statement before transmitting it to the node.

Pre-insert CQL

Cassandra Output gives you the option of executing an arbitrary set of CQL statements prior to inserting the first incoming PDI row. This is useful, amongst other things, for creating or dropping secondary indexes on columns.

Note: Pre-insert CQL statements are executed *after* any column family metadata updates for new incoming fields, and before the first row is inserted. This allows for indexes to be created for columns corresponding new incoming fields.

Option	Definition
CQL to execute before inserting first row	Opens the CQL editor, where you can enter one or more semicolon-separated CQL statements to execute before data is inserted into the first row.

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.

Option	Description
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)
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 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

Option	Description
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. 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.
Add filenames to result	Adds filenames to result filenames list.

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>

Option	Description
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.
Positive match	Turns filters into positive mode when turned on; only lines that match this filter will be passed. Negative filters will take precedence, and be immediately discarded.

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.
Position	
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

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

Scientific Notation... In a pattern, the exponent character immediately followed by one or more digit characters indicates scientific notation (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. **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)
Specify Date time format	Allows you to specify the date time format from the list within the `Date time format` dropdown list..
Date time format	Dropdown list of date format options.
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 are 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.

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.
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.
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: • Number - Total number of significant figures in a number • String - total length of string • Date - length of printed output of the string (for exampl, 4 returns year)
Precision	The precision option depends on the field type as follows: • 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.

HBase Input

This step reads data from an HBase table according to user-defined column metadata.

Configure Query

This tab contains connection details and basic query information. You can configure a connection in one of two ways: either via a comma-separated list of hostnames where the zookeeper quorum reside, or via an **hbase-site.xml** (and, optionally, **hbase-default.xml**) configuration file. If both zookeeper and HBase XML configuration options are supplied, then the zookeeper takes precedence.

Option	Definition
Step name	The name of this step as it appears in the transformation workspace.
Zookeeper host(s)	Comma-separated list of hostnames for the zookeeper quorum.
URL to hbase-site.xml	Address of the hbase-site.xml file.
URL to hbase-default.xml	Address of the hbase-default.xml file.
HBase table name	The source HBase table to read from. Click **Get Mapped Table Names** to populate the drop-down list of possible table names.
Mapping name	A mapping to decode and interpret column values. Click **Get Mappings For the Specified Table** to populate the drop-down list of available mappings.
Start key value (inclusive) for table scan	A starting key value to retrieve rows from. This is inclusive of the value entered.
Stop key value (exclusive) for table scan	A stopping key value for the scan. This is exclusive of the value entered. Both fields or the stop key field may be left blank. If the stop key field is left blank, then all rows from (and including) the start key will be returned.
Scanner row cache size	The number of rows that should be cached each time a fetch request is made to HBase. Leaving this blank uses the default, which is to perform no caching; one row would be returned per fetch request. Setting a value in this field will increase performance (faster scans) at the expense of memory consumption.
#	The order of query limitation fields.
Alias	The name that the field will be given in the output stream.
Key	Indicates whether the field is the table's key field or not.
Column family	The column family in the HBase source table that the field belongs to.
Column name	The name of the column in the HBase table (family + column name uniquely identifies a column in the HBase table).
Type	The PDI data type for the field.
Format	A formatting mask to apply to the field.
Indexed values	Indicates whether the field has a predefined set of values that it can assume.
Get Key/Fields Info	Assuming the connection information is complete and valid, this button will populate the field list and display the name of the key.

Create/Edit Mappings

This tab creates or edits a mapping for a given HBase table. A mapping simply defines metadata about the values that are stored in the table. Since just about all information is stored as raw bytes in HBase, this allows PDI to decode values and execute meaningful comparisons for column-based result set filtering.

Option	Definition
HBase table name	Displays a list of table names. Connection information in the previous tab must be valid and complete in order for this drop-down list to populate.
Mapping name	Names of any mappings that exist for the table. This box will be empty if there are no mappings defined for the

Option	Definition
	selected table, in which case you can enter the name of a new mapping.
#	The order of the mapping operation.
Alias	The name you want to assign to the HBase table key. This is required for the table key column, but optional for non-key columns.
Key	Indicates whether or not the field is the table's key.
Column family	The column family in the HBase source table that the field belongs to. Non-key columns must specify a column family and column name.
Column name	The name of the column in the HBase table.
Type	Data type of the column. Key columns can be of type: String Integer Unsigned integer (positive only) Long Unsigned long (positive only) Date Unsigned date. Non-key columns can be of type: String, Integer, Long, Float, Double, Boolean, Date, BigNumber, Serializable, Binary.
Indexed values	String columns may optionally have a set of legal values defined for them by entering comma-separated data into this field.

Filter Result Set

This tab provides two fields that limit the range of key values returned by a table scan. Leaving both fields blank will result in all rows being retrieved from the source table.

Option	Definition
Match all / Match any	When multiple column filters have been defined, you have the option returning only those rows that match all filters, or any single filter. Bounded ranges on a single numeric column can be defined by defining two filters (upper and lower bounds) and selecting **Match all**; similarly, open-ended ranges can be defined by selecting **Match any**.
#	The order of the filter operation.
Alias	A drop-down box of column alias names from the mapping.
Type	Data type of the column. This is automatically populated when you select a field after choosing the alias.
Operator	A drop-down box that contains either equality/inequality operators for numeric, date, and boolean fields; or substring and regular expression operators for string fields.
Comparison value	A comparison constant to use in conjunction with the operator.
Format	A formatting mask to apply to the field.
Signed comparison	Specifies whether or not the comparison constant and/or field values involve negative numbers (for non-string fields only). If field values and comparison constants are only positive for a given filter, then HBase's native lexicographical byte-based comparisons are sufficient. If this is not the case, then it is necessary for column values to be deserialized from bytes to actual numbers before performing the comparison.

Performance Considerations

Specifying fields in the Configure query tab will result in scans that return just those columns. Since HBase is a sparse column-oriented database, this requires that HBase check to see whether each row contains a specific column. More lookups equate to reduced speed, although the use of Bloom filters (if enabled on the table in question) mitigates this to a certain extent. If, on the other hand, the fields table in the Configure query tab is left blank, it results in a scan that returns rows that contain all columns that exist in each row (not only those that have been defined in the mapping). However, the HBase Input step will only emit those columns that are defined in the mapping being used. Because all columns are returned, HBase does not have to do any lookups. However, if the table in question contains many columns and is dense, then this will result in more data being transferred over the network.

HBase Output

This step writes data to an HBase table according to user-defined column metadata.

Configure Connection

This tab contains HBase connection information. You can configure a connection in one of two ways: either via a comma-separated list of hostnames where the zookeeper quorum reside, or via an **hbase-site.xml** (and, optionally, **hbase-default.xml**) configuration file. If both zookeeper and HBase XML configuration options are supplied, then the zookeeper takes precedence.

Option	Definition
Step name	The name of this step as it appears in the transformation workspace.
Zookeeper host(s)	Comma-separated list of hostnames for the zookeeper quorum.
URL to hbase-site.xml	Address of the hbase-site.xml file.
URL to hbase-default.xml	Address of the hbase-default.xml file.
HBase table name	The HBase table to write to. Click **Get Mapped Table Names** to populate the drop-down list of possible table names.
Mapping name	A mapping to decode and interpret column values. Click **Get Mappings For the Specified Table** to populate the drop-down list of available mappings.
Disable write to WAL	Disables writing to the Write Ahead Log (WAL). The WAL is used as a lifeline to restore the status quo if the server goes down while data is being inserted. Disabling WAL will increase performance.
Size of write buffer (bytes)	The size of the write buffer used to transfer data to HBase. A larger buffer consumes more memory (on both the client and server), but results in fewer remote procedure calls. The default (in the hbase-default.xml) is 2MB (2097152 bytes), which is the value that will be used if the field is left blank.

Create/Edit Mappings

This tab creates or edits a mapping for a given HBase table. A mapping simply defines metadata about the values that are stored in the table. Since just about all information is stored as raw bytes in HBase, this allows PDI to decode values and execute meaningful comparisons for column-based result set filtering.

Note: The names of fields entering the step are expected to match the aliases of fields defined in the mapping. All incoming fields must have a matching counterpart in the mapping. There may be fewer incoming fields than defined in the mapping, but if there are more incoming fields then an error will occur. Furthermore, one of the incoming fields must match the key defined in the mapping.

Option	Definition
HBase table name	Displays a list of table names. Connection information in the previous tab must be valid and complete in order for this drop-down list to populate.
Mapping name	Names of any mappings that exist for the table. This box will be empty if there are no mappings defined for the selected table, in which case you can enter the name of a new mapping.
#	The order of the mapping operation.

Option	Definition
Alias	The name you want to assign to the HBase table key. This is required for the table key column, but optional for non-key columns.
Key	Indicates whether or not the field is the table's key.
Column family	The column family in the HBase source table that the field belongs to. Non-key columns must specify a column family and column name.
Column name	The name of the column in the HBase table.
Type	Data type of the column. Key columns can be of type: String Integer Unsigned integer (positive only) Long Unsigned long (positive only) Date Unsigned date. Non-key columns can be of type: String, Integer, Long, Float, Double, Boolean, Date, BigNumber, Serializable, Binary.
Indexed values	String columns may optionally have a set of legal values defined for them by entering comma-separated data into this field.
Get incoming fields	Retrieves a field list using the given HBase table and mapping names.

Performance Considerations

The **Configure connection** tab provides a field for setting the size of the write buffer used to transfer data to HBase. A larger buffer consumes more memory (on both the client and server), but results in fewer remote procedure calls. The default (defined in the hbase-default.xml file) is 2MB. When left blank, the buffer is 2MB, **auto flush** is enabled, and **Put** operations are executed immediately. This means that each row will be transmitted to HBase as soon as it arrives at the step. Entering a number (even if it is the same as the default) for the size of the write buffer will disable auto flush and will result in incoming rows only being transferred once the buffer is full.

There is also a checkbox for disabling writing to the **Write Ahead Log** (WAL). The WAL is used as a lifeline to restore the status quo if the server goes down while data is being inserted. However, the tradeoff for error-recovery is speed.

The **Create/edit mappings** tab has options for creating new tables. In the **HBase table name** field, you can suffix the name of the new table with parameters for specifying what kind of compression to use, and whether or not to use Bloom filters to speed up lookups. The options for compression are: NONE, GZ and LZO; the options for Bloom filters are: NONE, ROW, ROWCOL. If nothing is selected (or only the name of the new table is defined), then the default of NONE is used for both compression and Bloom filters. For example, the following string entered in the HBase table name field specifies that a new table called "NewTable" should be created with GZ compression and ROWCOL Bloom filters:

```
NewTable@GZ@ROWCOL
```

Note: Due to licensing constraints, HBase does not ship with LZO compression libraries; these must be manually installed on each node if you want to use LZO compression.

MapReduce 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 MapReduce terms.
Value field	The Hadoop input field and data type that represents the **value** in MapReduce terms.

MapReduce 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 a 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 MapReduce terms.
Value field	The Hadoop output field that represents the **value** in MapReduce terms.

MongoDb Input

Options

Option	Definition
Step name	The name of this step as it appears in the transformation workspace.
Host name or IP address	The location of the MongoDB server. Check mongodb.log to see which IP address was used to run.
Port	The TCP/IP port to connect to, the default is 27017 (also check mongodb.log for the actual value).
Database	The name of the database to retrieve data from. Use the "show dbs" command when connected using the "mongodb" utility to list all databases on the server.
Collection	The name of the collection to retrieve data from. Use the "show collections" command when connected using the "mongodb" utility to list all collections in the database.
Name of the JSON output field	The name of the field that will contain the JSON output from the server. You can parse this JSON then using the "JSON Input" step, eval("{"+jsonString +"}"), in JavaScript or using a *User Defined Java Class step*.
Query expression (JSON)	The query expression in JSON that will limit the output (see examples below).

Option	Definition
Authentication user	The user to use for the connection with MongoDB.
Authentication password	The password to use.

Query Examples

The *Advanced Queries* page in the MongoDB wiki space details how to use queries. What is not mentioned is that in order for us to pass these queries to MongoDB using the Java API (on which PDI is built) we need to add appropriate quoting. Below are some translated examples:

Query expression	Description
{ 'name' : "MongoDB" }	Query all values where the name field equals to "MongoDB".
{ 'name' : { '$regex' : 'm.*', '$options' : "i" } }	Uses a regular expression to find names starting with m, case insensitive.
{ 'name' : { '$gt' : "M" } }	Searches all strings greater than M.
{ 'name' : { '$lte' : "T" } }	Searches all strings less than or equal to "T".
{ 'name' : { '$in' : \["MongoDB", "MySQL"] } }	Finds all names that are either MongoDB or MySQL.
{ 'name' : { '$nin' : ["MongoDB", "MySQL"] } }	Finds all names that are either MongoDB or MySQL.
{ '$where' : "this.count == 1" }	Uses JavaScript to evaluate a condition.

MongoDb Output

MongoDb Output writes to a MongoDB collection.

Configure connection

Option	Definition
Step name	The name of this step as it appears in the transformation workspace.
Host name or IP address	The network name or address of the MongoDB instance
Port	Port number of the MongoDB instance
Username	If the database requires authentication, this is the user credential
Password	The password for the given username
Database	The name of the database to use. If you entered the correct connection and authentication information, you can click **Get DBs** to show a list of databases.
Collection	The collection you want to write to in the specified database. If you entered the correct connection and authentication information, you can click **Get collections** to show a list of collections. By default, data is inserted into the target collection. If the specified collection doesn't exist, it will be created before data is inserted.
Batch insert size	Mongo DB allows for fast bulk insert operations. This option sets the batch size. If left blank, the default size will be 100 rows.

Option	Definition
Truncate collection	Deletes any existing data in the target collection before inserting begins. MongoDB will allow duplicate records to be inserted unless you are using unique indexes.
Upsert	Changes the write mode from insert to upsert, which updates if a match is found, and inserts a new record if there is no match.
Multi-update	Updates all matching documents, rather than just the first.
Modifier update	Enables modifier ($) operators to be used to mutate individual fields within matching documents. This type of update is fast and involves minimal network traffic.

Mongo document fields

Option	Definition
#	The order of this field in the list
Name	The name of this field, descriptive of its content
Mongo document path	The hierarchical path to each field
Use field name	Specifies whether the incoming field name will be used as the final entry in the path. When this is set to **Y** for a field, a preceding **.** (dot) is assumed.
Match field for upsert	Specifies which of the fields should be used for matching when performing an upsert operation. The first document in the collection that matches all fields tagged as "Y" in this column is replaced with the new document constructed with incoming values for all of the defined field paths. If no matching document is found, then a new document is inserted into the collection.
Modifier operation	In-place modifications of existing document fields. These are much faster than replacing a document with a new one. It is also possible to update more than one matching document in this way. Selecting the **Modifier update** checkbox in conjunction with **Upsert** enables this mode of updating. Selecting the **Multi-update** checkbox as well enables each update to apply to all matching documents (rather than just the first). Valid modifier operations are: **$set**, **$inc**, and **$push**. It also supports the positional operator ($) for matching inside of arrays.
Get fields	Populates the left-hand column of the table with the names of the incoming fields
Preview document structure	Shows the structure that will be written to MongoDB in JSON format

Create/drop indexes

Option	Definition
#	The order of this field in the list
Index fields	Specifies a single index (using one field) or a compound index (using multiple fields). The **.** (dot) notation is used to specify a path to a field to use in the index. This path can be optionally postfixed by a direction indicator. Compound indexes are specified by a comma-separated list of paths.
Index opp	Specifies whether this index will be created or dropped
Unique	Makes the index unique
Sparse	Makes the index sparse
Show indexes	Shows the index information available

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 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.

Option	Description
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.

PDI Job Entry Reference

The job steps explained below pertain to Hadoop functions in Pentaho Data Integration. Some of them are Hadoop-specific, and others are standard PDI steps that are required for some kinds of Hadoop jobs.

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 Executer step instance.
EMR Job Flow Name	The name of the Amazon EMR job flow (series of steps) you are executing.
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 MapReduce 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.

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 Executer step instance.

Option	Definition
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.
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.

Option	Definition
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.

Pentaho MapReduce

Note: This entry was formerly known as **Hadoop Transformation Job Executor**.

This job entry executes transformations as part of a Hadoop MapReduce job. 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. Any properties defined here will be set in the MapReduce job configuration.

General

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

Mapper

Option	Definition
Look in	Sets the context for the Browse button. Options are: **Local** (the local filesystem), **Repository by Name** (a PDI database or enterprise repository), or **Repository by Reference** (a link to a transformation no matter which repository it is in).
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 a MapReduce Input step.
Mapper Output Step Name	The name of the step that passes mapping output back to Hadoop. This must be a MapReduce Output step.

Combiner

Option	Definition
Look in	Sets the context for the Browse button. Options are: **Local** (the local filesystem), **Repository by Name** (a PDI database or enterprise repository), or **Repository by Reference** (a link to a transformation no matter which repository it is in).
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 a MapReduce Input step.
Combiner Output Step Name	The name of the step that passes combiner output back to Hadoop. This must be a MapReduce Output step.
Combine single threaded	Indicates if the Single Threaded transformation execution engine should be used to execute the combiner transformation. If false, the normal multi-threaded transformation engine will be used. The Single Threaded transformation execution engine reduces overhead when processing many small groups of output.

Reducer

Option	Definition
Look in	Sets the context for the Browse button. Options are: **Local** (the local filesystem), **Repository by Name** (a PDI database or enterprise repository), or **Repository by Reference** (a link to a transformation no matter which repository it is in).
Reducer Transformation	The KTR that will perform the reducer functions for this job.

Option	Definition
Reducer Input Step Name	The name of the step that receives reducing data from Hadoop. This must be a MapReduce Input step.
Reducer Output Step Name	The name of the step that passes reducing output back to Hadoop. This must be a MapReduce Output step.
Reduce single threaded	Indicates if the Single Threaded transformation execution engine should be used to execute the reducer transformation. If false, the normal multi-threaded transformation engine will be used. The Single Threaded transformation execution engine reduces overhead when processing many small groups of output.

Job Setup

Option	Definition
Suppress Output of Map Key	If selected the key output from the Mapper transformation will be ignored and replaced with NullWritable.
Suppress Output of Map Value	If selected the value output from the Mapper transformation will be ignored and replaced with NullWritable.
Suppress Output of Reduce Key	If selected the key output from the Combiner and/or Reducer transformations will be ignored and replaced with NullWritable.
Suppress Output of Reduce Value	If selected the key output from the Combiner and/or Reducer transformations will be ignored and replaced with NullWritable.
Input Path	A comma-separated list of input directories from your Hadoop cluster that will be used when using a file-based input format derived from *FileInputFormat*.
Output Path	The directory output from the MapReduce should be written to when using a file-based output format derived from *FileOutputFormat*.
Input Format	The Apache Hadoop class name that describes the input specification for the MapReduce job. See *InputFormat* for more information.
Output Format	The Apache Hadoop class name that describes the output specification for the MapReduce job. See *OutputFormat* for more information.
Clean output path before execution	If enabled the output path specified will be removed before the MapReduce job is scheduled.

Cluster

Option	Definition
Hadoop Distribution	The Hadoop Distribution to connect to. Only Cloudera has a different option; all others are **generic**.
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 becomes the output of the entire job; 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 job entry. Error handling/ routing will not work unless this option is checked.
Logging Interval	Number of seconds between log messages.

Pig Script Executor

Executes a script written in Apache Pig's "Pig Latin" language on a Hadoop cluster.

Note: All log entries pertaining to this script execution that are generated by Apache Pig will show in the PDI log.

General

Option	Definition
Job Entry Name	The name of this Pig Script Executor instance.
HDFS hostname	The hostname of the machine that operates a Hadoop distributed filesystem.
HDFS port	The port number of the machine that operates a Hadoop distributed filesystem.
Job tracker hostname	The hostname of the machine that operates a Hadoop job tracker.
Job tracker port	The port number of the machine that operates a Hadoop job tracker.
Pig script	The path (remote or local) to the Pig Latin script you want to execute.
Enable blocking	If checked, the Pig Script Executor job entry will prevent downstream entries from executing until the script has finished processing.
Local execution	Executes the script within the same Java virtual machine that PDI is running in. This option is useful for testing and debugging because it does not require access to a Hadoop cluster. When this option is selected, the HDFS and job tracker connection details are not required and their corresponding fields will be disabled.

Script Parameters

Option	Definition
#	The order of execution of the script parameters.
Parameter name	The name of the parameter you want to use.
Value	The value you're substituting whenever the previously defined parameter is used.

Hadoop to PDI Data Type Conversion

The Hadoop Job Executor and Pentaho MapReduce 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

Hadoop Hive-Specific SQL Limitations

There are a few key limitations in Hive that prevent some regular Metadata Editor features from working as intended, and will limit the structure of your SQL queries in Report Designer:

- **Outer joins are not supported.**
- **Each column can only be used once in a SELECT clause.** Duplicate columns in SELECT statements will cause errors.
- **Conditional joins can only use the = conditional unless you use a WHERE clause.** Any non-equal conditional in a FROM statement will force Metadata Editor to use a cartesian join and a WHERE clause conditional to limit it. This is not much of a limitation, but it will seem unusual to experienced Metadata Editor users who are accustomed to working with SQL databases.

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 Business Analytics 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 BA Server):

Note: For the DI 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.

- **BA 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 BA 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.

Adding a JDBC Driver to Hadoop

You must ensure that your Hadoop nodes have a JDBC driver JAR for every database they will connect to. If you are missing any drivers, copy the JAR files to the `/lib/` subdirectory in your Hadoop home.

Note: The Pentaho Data Integration client tools come with many common JDBC drivers in the `/pentaho/design-tools/data-integration/libext/JDBC/` directory that you can use in Hadoop.

```
cp /tmp/downloads/mysql-connector-java-3.1.14-bin.jar /hadoop-0.20.2/lib/
```