



Work with Big Data

Copyright Page

This document supports Pentaho Business Analytics Suite 5.3 GA and Pentaho Data Integration 5.3 GA, documentation revision January 15th, 2015, copyright © 2015 Pentaho Corporation. No part may be reprinted without written permission from Pentaho Corporation. All trademarks are the property of their respective owners.

Help and Support Resources

To view the most up-to-date help content, visit <https://help.pentaho.com>.

If you do not find answers to your questions here, 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, visit <http://www.pentaho.com/training>.

Liability Limits and Warranty Disclaimer

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

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

Trademarks

The trademarks, logos, and service marks ("Marks") displayed on this website are the property of Pentaho Corporation or third party owners of such Marks. You are not permitted to use, copy, or imitate the Mark, in whole or in part, without the prior written consent of Pentaho Corporation or such third party. Trademarks of Pentaho Corporation include, but are not limited, to "Pentaho", its products, services and the Pentaho logo.

Trademarked names may appear throughout this website. Rather than list the names and entities that own the trademarks or inserting a trademark symbol with each mention of the trademarked name, Pentaho Corporation states that it is using the names for editorial purposes only and to the benefit of the trademark owner, with no intention of infringing upon that trademark.

Third-Party Open Source Software

For a listing of open source software used by each Pentaho component, navigate to the folder that contains the Pentaho component. Within that folder, locate a folder named licenses. The licenses folder contains HTML files that list the names of open source software, their licenses, and required attributions.

Contact Us

Global Headquarters Pentaho Corporation Citadel International, Suite 460

5950 Hazeltine National Drive Orlando, FL 32822

Phone: +1 407 812-OPEN (6736)

Fax: +1 407 517-4575

<http://www.pentaho.com>

Sales Inquiries: sales@pentaho.com

Getting Started with PDI and Hadoop

Pentaho provides a complete big data analytics solution that supports the entire big data analytics process. From big data aggregation, preparation, and integration, to interactive visualization, analysis, and prediction, Pentaho allows you to harvest the meaningful patterns buried in big data stores. Analyzing your big data sets gives you the ability to identify new revenue sources, develop loyal and profitable customer relationships, and run your organization more efficiently and cost effectively.

- [Pentaho, Big Data, and Hadoop](#)
- [About Hadoop](#)
- [Big Data Resources](#)

Pentaho, Big Data, and Hadoop

The term big data applies to very large, complex, or dynamic datasets that need to be stored and managed over a long time. To derive benefits from big data, you need the ability to access, process, and analyze data as it is being created. However, the size and structure of big data makes it very inefficient to maintain and process it using traditional relational databases.

Big data solutions re-engineer the components of traditional databases—data storage, retrieval, query, processing—and massively scales them.

Pentaho Big Data Overview

Pentaho increases speed-of-thought analysis against even the largest of big data stores by focusing on the features that deliver performance.

- **Instant access**—Pentaho provides visual tools to make it easy to define the sets of data that are important to you for interactive analysis. These data sets and associated analytics can be easily shared with others, and as new business questions arise, new views of data can be defined for interactive analysis.
- **High performance platform**—Pentaho is built on a modern, lightweight, high performance platform. This platform fully leverages 64-bit, multi-core processors and large memory spaces to efficiently leverage the power of contemporary hardware.
- **Extreme-scale, in-memory caching**—Pentaho is unique in leveraging external data grid technologies, such as Infinispan and Memcached to load vast amounts of data into memory so that it is instantly available for speed-of-thought analysis.
- **Federated data integration**—Data can be extracted from multiple sources, including big data and traditional data stores, integrated together and then flowed directly into reports, without needing an enterprise data warehouse or data mart.

About Hadoop

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

A Hadoop platform consists of a Hadoop kernel, a [MapReduce](#) model, a distributed file system, and often a number of related projects—such as [Apache Hive](#), [Apache HBase](#), and others.

A Hadoop Distributed File System, commonly referred to as HDFS, is a Java-based, distributed, scalable, and portable file system for the Hadoop framework.

Big Data Resources

- [Pentaho Big Data Analytics Center](#)
- [Pentaho Big Data Wiki](#)
- [Apache Hadoop project](#) -- A project that contains libraries that allows for the distributed processing of large data sets across clusters of computers using simple programming models. There are several modules, including the [Hadoop Distributed File System \(HDFS\)](#), which is a distributed file system that provides high-throughput access to application data and [Hadoop MapReduce](#), which is a key algorithm to distribute work around a cluster.
- [Avro](#)—A data serialization system
- [Cassandra](#)—A scalable multi-master database with no single points of failure
- [HBase](#)—A scalable, distributed database that supports structured data storage for large tables
- [Hive](#)—A data warehouse infrastructure that provides data summarization and on-demand querying
- [Pig](#)—A high-level, data-flow language and execution framework for parallel computation
- [ZooKeeper](#)—A high-performance coordination service for distributed applications
- [MongoDB](#)— A NoSQL open source document-oriented database system developed and supported by 10gen
- [Splunk](#) - A data collection, visualization and indexing engine for operational intelligence that is developed by Splunk, Inc.
- [CouchDB](#)—A NoSQL open source document-oriented database system developed and supported by Apache
- [Sqoop](#)—Software for transferring data between relational databases and Hadoop
- [Oozie](#)—A workflow scheduler system to manage Hadoop jobs

Configure Your Big Data Environment

Configuring a Pentaho component such as Spoon, DI Server, BA Server, PRD, Metadata Editor is easy. Pentaho supports many different Hadoop distributions including Cloudera, MapR, Hortonworks, DataStax, and Apache.

To configure the Pentaho, do two things.

- Get the Hadoop distribution you want to use
- Set the active Hadoop distribution

For instructions on how to do these things, and to see which Hadoop Distributions we support, see [Configure Pentaho for Your Hadoop Distribution and Version](#) on the Pentaho Big Data Wiki.

Working with Big Data and Hadoop in PDI

Pentaho Data Integration (PDI) can operate in two distinct modes, job orchestration and data transformation. Within PDI they are referred to as jobs and transformations.

PDI jobs sequence a set of entries that encapsulate actions. An example of a PDI big data job would be to check for existence of new log files, copy the new files to HDFS, execute a MapReduce task to aggregate the weblog into a click stream and stage that clickstream data in an analytic database.

PDI transformations consist of a set of steps that execute in parallel and operate on a stream of data columns. The columns usually flow from one system, through the PDI engine, where new columns can be calculated or values can be looked up and added to the stream. The data stream is then sent to a receiving system like a Hadoop cluster, a database, or even the Pentaho Reporting Engine.

The tutorials within this section illustrate how to use PDI jobs and transforms in typical big data scenarios. PDI job entries and transformation steps are described in the [Transformation Step Reference](#) and [Job Entry Reference](#) sections of Administer the DI Server.

PDI's Big Data Plugin

The Pentaho Big Data plugin contains all of the job entries and transformation steps required for working with Hadoop, Cassandra, and MongoDB.

By default, PDI is pre-configured to work with Apache Hadoop 0.20.X. But PDI can be configured to communicate with most popular Hadoop distributions. Instructions for changing Hadoop configurations are covered in the [Configure Your Big Data Environment](#) section.

For a list of supported big data technology, including which configurations of Hadoop are currently supported, see the section on [Supported Components](#).

Using PDI Outside and Inside the Hadoop Cluster

PDI is unique in that it can execute both outside of a Hadoop cluster and within the nodes of a hadoop cluster. From outside a Hadoop cluster, PDI can extract data from or load data into Hadoop HDFS, Hive and HBase. When executed within the Hadoop cluster, PDI transformations can be used as Mapper and/or Reducer tasks, allowing PDI with Pentaho MapReduce to be used as visual programming tool for MapReduce.

These videos demonstrate using PDI to work with Hadoop from both inside and outside a Hadoop cluster.

- Loading Data into Hadoop from outside the Hadoop cluster is a 5-minute video that demonstrates moving data using a PDI job and transformation: <http://www.youtube.com/watch?v=Ylekzmd6TAc>

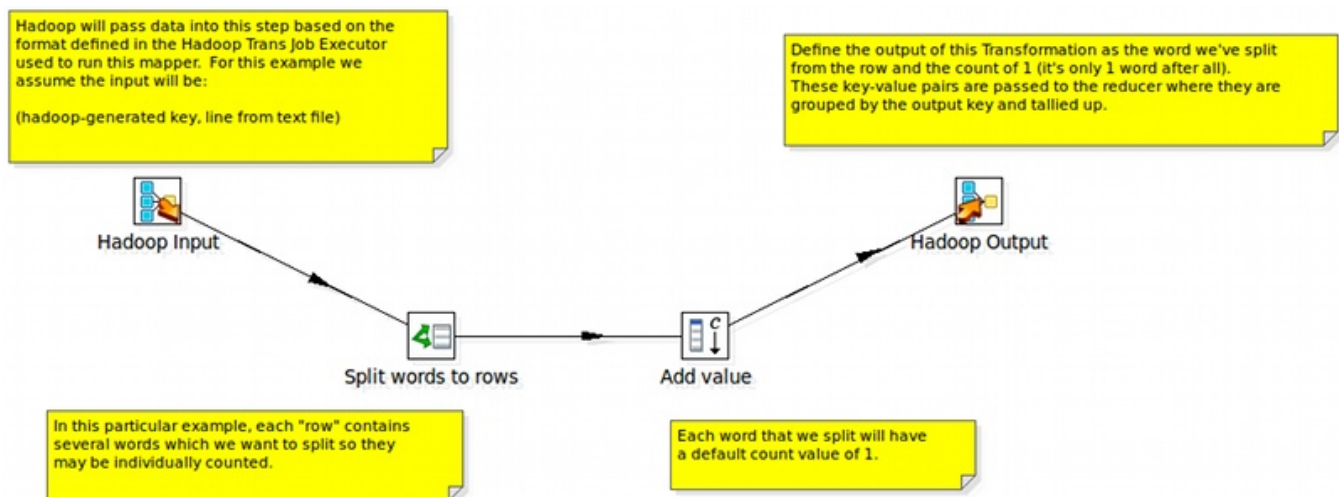
- Use [Pentaho MapReduce](#) to interactively design a data flow for a MapReduce job without writing scripts or code. Here is a 12 minute video that provides an overview of the process: <http://www.youtube.com/watch?v=KZe1UugxXcs>.
- [Pentaho MapReduce Workflow](#)
- [PDI Hadoop Job Workflow](#)
- [Hadoop to PDI Data Type Conversion](#)
- [Hadoop Hive-Specific SQL Limitations](#)
- [Big Data Tutorials](#)

Pentaho MapReduce Workflow

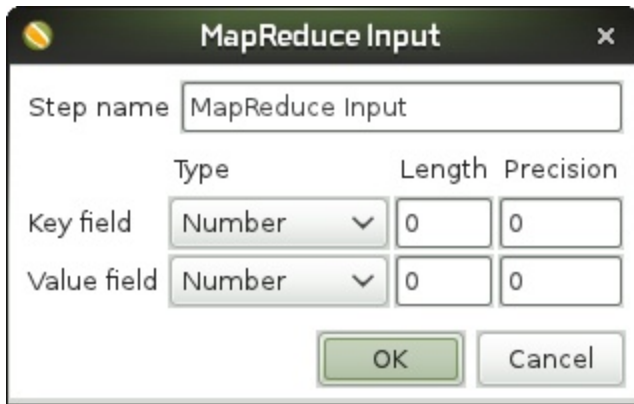
PDI and Pentaho MapReduce enables you to pull data from a Hadoop cluster, transform it, and pass it back to the cluster. Here is how you would approach doing this.

PDI Transformation

Start by deciding what you want to do with your data, open a PDI transformation, and drag the appropriate steps onto the canvas, configuring the steps to meet your data requirements. Drag the specifically-designed Hadoop **MapReduce Input** and Hadoop **MapReduce Output** steps onto the canvas. PDI provides these steps to completely avoid the need to write Java classes for this functionality. Configure both of these steps as needed. Once you have configured all the steps, add hops to sequence the steps as a transformation. Follow the workflow as shown in this sample transformation in order to properly communicate with Hadoop. Name this transformation Mapper.



Hadoop communicates in key/value pairs. PDI uses the **MapReduce Input** step to define how key/value pairs from Hadoop are interpreted by PDI. The **MapReduce Input** dialog box enables you to configure the **MapReduce Input** step.



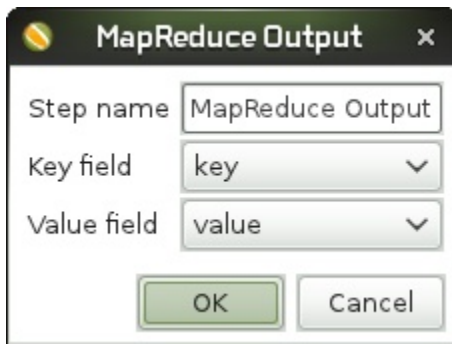
MapReduce Input

Step name: MapReduce Input

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

OK Cancel

PDI uses a **MapReduce Output** step to pass the output back to Hadoop. The **MapReduce Output** dialog box enables you to configure the **MapReduce Output** step.



MapReduce Output

Step name: MapReduce Output

Key field: key

Value field: value

OK Cancel

What happens in the middle is entirely up to you. Pentaho provides many sample steps you can alter to create the functionality you need.

PDI Job

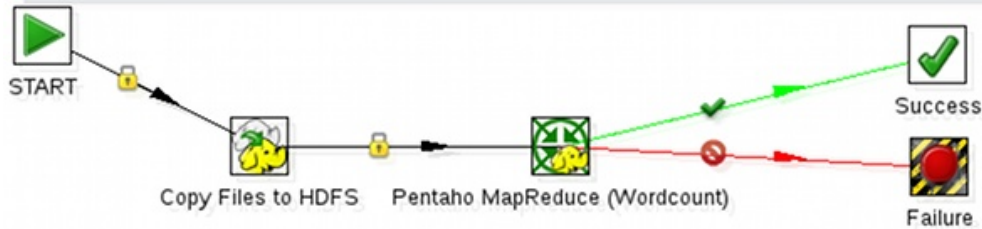
Once you have created the Mapper transformation, you are ready to include it in a **Pentaho MapReduce** job entry and build a MapReduce job. Open a PDI job and drag the specifically-designed **Pentaho MapReduce** job entry onto the canvas. In addition to ordinary transformation work, this entry is designed to execute mapper/reducer functions within PDI. Again, no need to provide a Java class to achieve this.

Configure the **Pentaho MapReduce** entry to use the transformation as a mapper. Drag and drop a Start job entry, other job entries as needed, and result jobentries to handle the output onto the canvas. Add hops to sequence the entries into a job that you execute in PDI.

The workflow for the job should look 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



The Pentaho MapReduce dialog box enables you to configure the Pentaho MapReduce entry.

Pentaho MapReduce

Name: Pentaho MapReduce

Hadoop Job Name:

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

Look in: Local

Mapper Transformation: Browse...

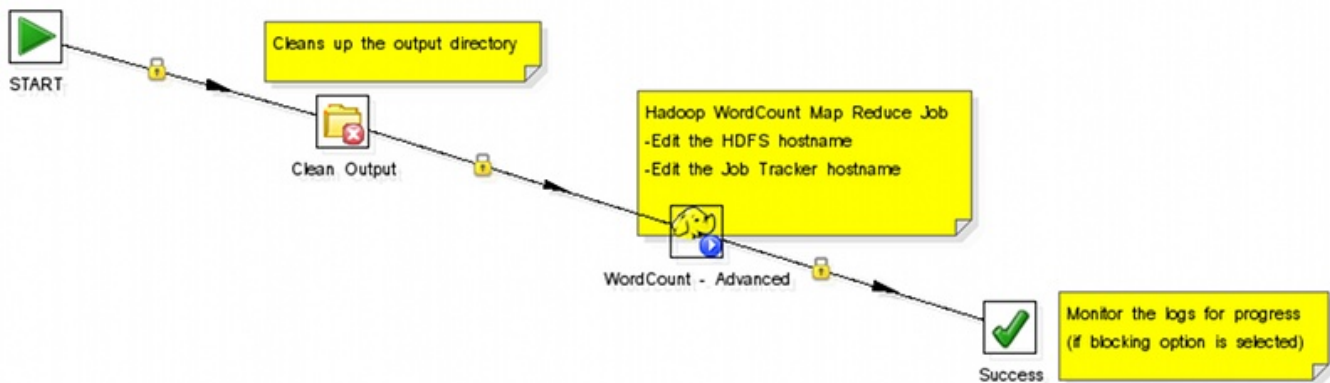
Mapper Input Step Name:

Mapper Output Step Name:

OK Cancel

PDI Hadoop Job Workflow

PDI enables you to execute a Java class from within a PDI/Spoon job to perform operations on Hadoop data. The way you approach doing this is similar to the way you would for any other PDI job. The specifically-designed job entry that handles the Java class is **Hadoop Job Executor**. In this illustration it is used in the **WordCount - Advanced** entry.



The **Hadoop Job Executor** dialog box enables you to configure the entry with a `jar` file that contains the Java class.

The screenshot shows a dialog box titled "Hadoop Job Executor" with a close button (X) in the top right corner. The dialog contains several input fields and a configuration section. The "Name:" field is filled with "WordCount - Simple". The "Hadoop Job Name:" field is filled with "PDI Hadoop - WordCount - Simple". The "Jar:" field is filled with ".\samples\jobs\hadoop\pentaho-mapreduce-sample.jar" and has a "Browse..." button to its right. Below these fields is a "Configuration" section with two radio buttons: "Simple" (which is selected) and "Advanced". At the bottom of the dialog is a "Command line arguments:" field filled with "/junit/wordcount/input /junit/wordcount/pdioutput". In the bottom right corner, there are "OK" and "Cancel" buttons.

Hadoop Job Executor

Name: WordCount - Simple

Hadoop Job Name: PDI Hadoop - WordCount - Simple

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

Configuration

☒ Simple ☐ Advanced

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

OK Cancel

If you are using the Amazon Elastic MapReduce (EMR) service, you can **Amazon EMR Job Executor**. job entry to execute the Java class 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

OKCancel

Hadoop to PDI Data Type Conversion

The **Hadoop Job Executor** and **Pentaho MapReduce** steps have an advanced configuration mode that enables 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. This table explains the relationship between Hadoop data types and their PDI equivalents.

PDI (Kettle) Data Type	Apache Hadoop Data Type
<code>java.lang.Integer</code>	<code>org.apache.hadoop.io.IntWritable</code>
<code>java.lang.Long</code>	<code>org.apache.hadoop.io.IntWritable</code>
<code>java.lang.Long</code>	<code>org.apache.hadoop.io.LongWritable</code>
<code>org.apache.hadoop.io.IntWritable</code>	<code>java.lang.Long</code>
<code>java.lang.String</code>	<code>org.apache.hadoop.io.Text</code>
<code>java.lang.String</code>	<code>org.apache.hadoop.io.IntWritable</code>
<code>org.apache.hadoop.io.LongWritable</code>	<code>org.apache.hadoop.io.Text</code>
<code>org.apache.hadoop.io.LongWritable</code>	<code>java.lang.Long</code>

For more information on configuring **Pentaho MapReduce** to convert to additional data types, see <http://wiki.pentaho.com/display/BAD/Pentaho+MapReduce>.

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 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 cause errors.
- **Conditional joins can only use the = conditional unless you use a WHERE clause.** Any non-equal conditional in a FROM statement forces the Metadata Editor to use a cartesian join and a WHERE clause conditional to limit it. This is not much of a limitation, but it may seem unusual to experienced Metadata Editor users who are accustomed to working with SQL databases.

Big Data Tutorials

These sections contain guidance and instructions about using Pentaho technology as part of your overall big data strategy. Each section is a series of scenario-based tutorials that demonstrate the integration between Pentaho and Hadoop using a sample data set.

- [Hadoop Tutorials](#)
- [MapR Tutorials](#)
- [Cassandra Tutorials](#)
- [MongoDB Tutorials](#)

Hadoop Tutorials

These tutorials are organized by topic and each set explains various techniques for loading, transforming, extracting and reporting on data within a Hadoop cluster. You are encouraged to perform the tutorials in order as the output of one is sometimes used as the input of another. However, if you would like to jump to a tutorial in the middle of the flow, instructions for preparing input data are provided.

- [Loading Data into a Hadoop Cluster](#)
- [Transforming Data within a Hadoop Cluster](#)
- [Extracting Data from a Hadoop Cluster](#)
- [Reporting on Data within a Hadoop Cluster](#)

Loading Data into a Hadoop Cluster

These scenario-based tutorials contain guidance and instructions on loading data into HDFS (Hadoop's Distributed File System), Hive and HBase using Pentaho Data Integration (PDI)

- [Prerequisites](#)
- [Using a Job Entry to Load Data into Hadoop's Distributed File System \(HDFS\)](#)
- [Using a Job Entry to Load Data into Hive](#)
- [Using a Transformation Step to Load Data into HBase](#)

Prerequisites

To perform the tutorials in this section you must have these components installed.

PDI—The primary development environment for the tutorials. See the [Data Integration Installation Options](#) if you have not already installed PDI.

Apache Hadoop 0.20.X—A single-node local cluster is sufficient for these exercises, but a larger and/or remote configuration also works. If you are using a different distribution of Hadoop see [Configure Your Big Data Environment](#). You need to know the addresses and ports for your Hadoop installation.

***Hive**—A supported version of Hive. Hive is a Map/Reduce abstraction layer that provides SQL-like access to Hadoop data. For instructions on installing or using Hive, see the [Hive Getting Started Guide](#).

***HBase**—A supported version of HBase. HBase is an open source, non-relational, distributed database that runs on top of HDFS. For instructions on installing or using HBase, see the [Getting Started section of the Apache HBase Reference Guide](#).

**Component only required for corresponding tutorial.*

- [Sample Data](#)

Sample Data

The tutorials in this section were created with this sample weblog data.

Tutorial	File Name	Content
Using a Job Entry to Load Data into Hadoop's Distributed File System (HDFS)	weblogs_rebuild.txt.zip	Unparsed, raw weblog data
Using a Job Entry to Load Data into Hive	weblogs_parse.txt.zip	Tab-delimited, parsed weblog data
Using a Transformation Step to Load Data into HBase	weblogs_hbase.txt.zip	Prepared data for HBase load

Using a Job Entry to Load Data into Hadoop's Distributed File System (HDFS)

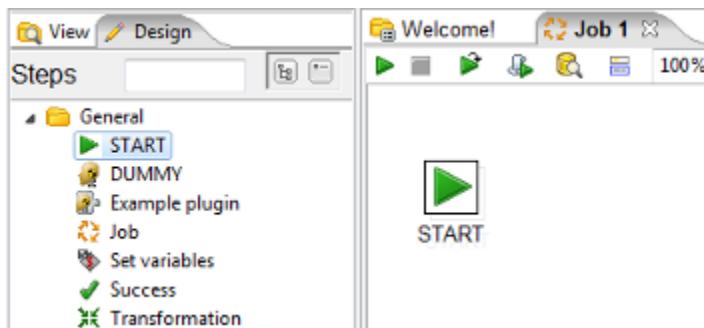
In order to follow along with this tutorial, you will need

- Hadoop
- Pentaho Data Integration

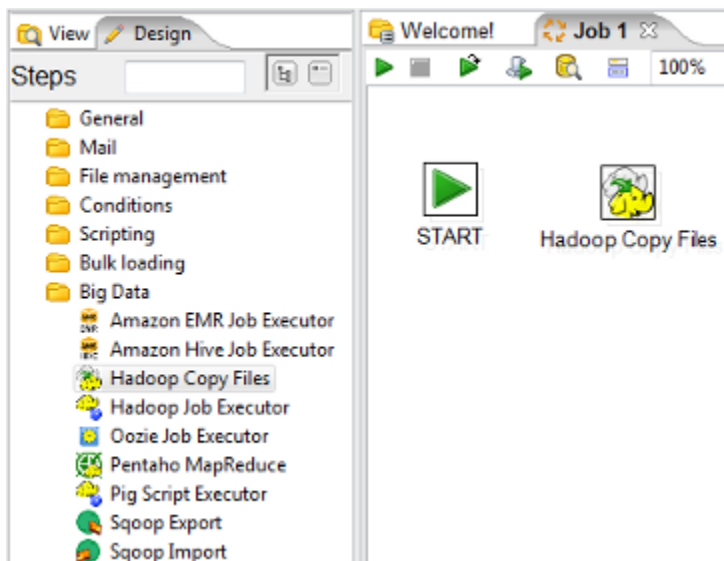
You can use PDI jobs to put files into HDFS from many different sources. This tutorial describes how to create a PDI job to move a sample file into HDFS.

If not already running, start Hadoop and PDI. Unzip the sample data files and put them in a convenient location: [weblogs_rebuild.txt.zip](#).

1. Create a new Job by selecting **File > New > Job**.
2. Add a Start job entry to the canvas. From the **Design** palette on the left, under the **General** folder, drag a **Start** job entry onto the canvas.



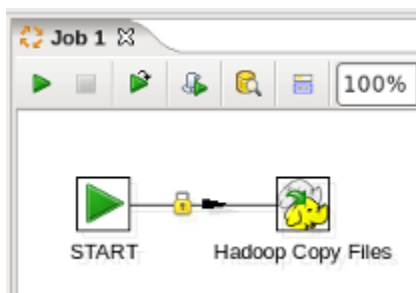
3. Add a Hadoop Copy Files job entry to the canvas. From the **Design** palette, under the **Big Data** folder, drag a **Hadoop Copy Files** job entry onto the canvas.



4. Connect the two job entries by hovering over the **Start** entry and selecting the output connector

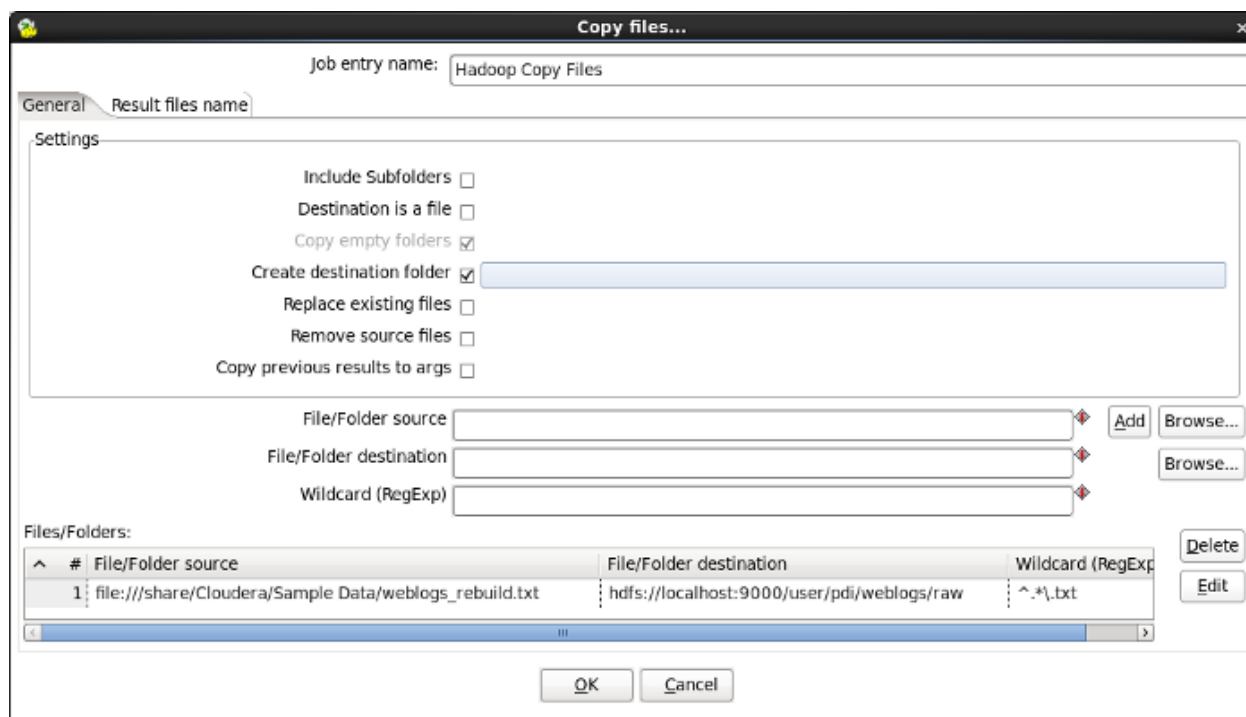


, then drag the connector arrow to the **Hadoop Copy Files** entry.



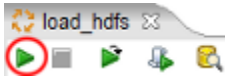
5. Enter the source and destination information within the properties of the **Hadoop Copy Files** entry by double-clicking it.
 - a. For **File/Folder source(s)**, click **Browse** and navigate to the folder containing the downloaded sample file `weblogs_rebuild.txt`.
 - b. For **File/Folder destination(s)**, enter `hdfs://<NAMENODE>:<PORT>/user/pdi/weblogs/raw`, where `NAMENODE` and `PORT` reflect your Hadoop destination.
 - c. For **Wildcard (RegExp)**, enter `^.*\.txt`.
 - d. Click **Add** to include the entries to the list of files to copy.
 - e. Check the **Create destination folder** option to ensure that the `weblogs` folder is created in HDFS the first time this job is executed.

When you are done your window should look like this (your file paths may be different).



Click **OK** to close the window.

6. Save the job by selecting **Save as** from the **File** menu. Enter `load_hdfs.kjb` as the file name within a folder of your choice.
7. Run the job by clicking the green Run button on the job toolbar



, or by selecting **Action > Run** from the menu. The **Execute a job** window opens. Click **Launch**.

An **Execution Results** panel opens at the bottom of the Spoon interface and displays the progress of the job as it runs. After a few seconds the job finishes successfully.

Execution results						
History Logging Job metrics						
Job / Job Entry	Comment	Result	Reason	Filename	Nr	Log date
load_hdfs						
Job: load_hdfs	Start of job execution		start			2012/01/27 08:25:45
START	Start of job execution		start			2012/01/27 08:25:45
START	Job execution finished	Success			0	2012/01/27 08:25:45
Hadoop Copy Files	Start of job execution		Followed unconditional link			2012/01/27 08:25:45
Hadoop Copy Files	Job execution finished	Success			0	2012/01/27 08:25:46
Job: load_hdfs	Job execution finished	Success	finished		0	2012/01/27 08:25:46

If any errors occurred the job entry that failed will be highlighted in red and you can use the **Logging** tab to view error messages.

8. Verify the data was loaded by querying Hadoop.
 - a. From the command line, query Hadoop by entering this command.

```
hadoop fs -ls /user/pdi/weblogs/raw
```

This statement is returned

```
-rwxrwxrwx 3 demo demo 77908174 2011-12-28 07:16 /user/pdi/weblogs/raw/weblog_raw.txt
```

Using a Job Entry to Load Data into Hive

In order to follow along with this tutorial, you will need

- Hadoop
- Pentaho Data Integration
- Hive

PDI jobs can be used to put files into Hive from many different sources. This tutorial instructs you how to use a PDI job to load a sample data file into a Hive table.

Note: Hive could be defined with external data. Using the external option, you could define a Hive table that uses the HDFS directory that contains the parsed file. For this tutorial, we chose not to use the external option to demonstrate the ease with which files can be added to non-external Hive tables.

If not already running, start Hadoop, PDI, and the Hive server. Unzip the sample data files and put them in a convenient location: [weblogs_parse.txt.zip](#).

This file should be placed in the `/user/pdi/weblogs/parse` directory of HDFS using these three commands.

```
hadoop fs -mkdir /user/pdi/weblogs
hadoop fs -mkdir /user/pdi/weblogs/parse
hadoop fs -put weblogs_parse.txt /user/pdi/weblogs/parse/part-00000
```

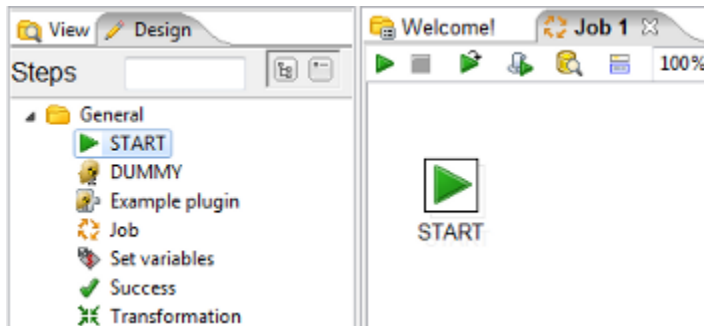
If you previously completed the [Using Pentaho MapReduce to Parse Weblog Data](#) tutorial, the necessary files will already be in the proper directory.

1. Create a Hive Table.
 - a. Open the Hive shell by entering `'hive'` at the command line.
 - b. Create a table in Hive for the sample data by entering

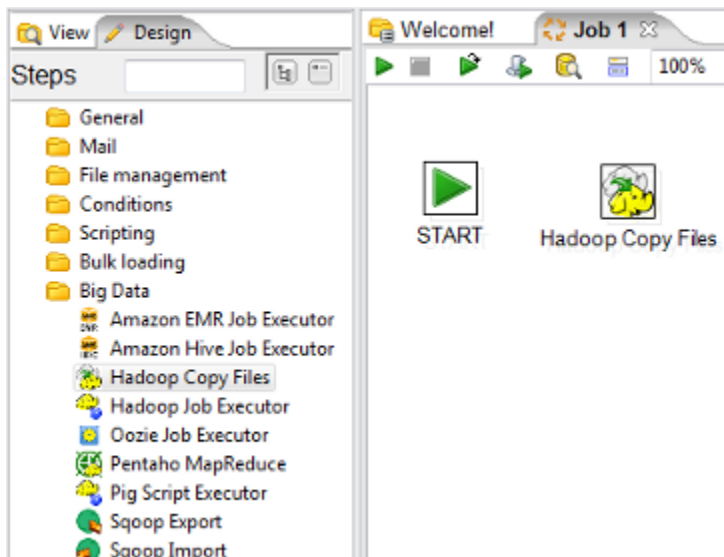
```
create table weblogs (
  client_ip      string,
  full_request_date string,
  day           string,
  month         string,
  month_num     int,
  year          string,
  hour          string,
  minute        string,
  second        string,
  timezone      string,
  http_verb     string,
```

```
uri      string,
http_status_code  string,
bytes_returned    string,
referrer          string,
user_agent        string)
row format delimited
fields terminated by '\t';
```

- c. Close the Hive shell by entering 'quit'.
2. Create a new Job to load the sample data into a Hive table by selecting **File > New > Job**.
3. Add a Start job entry to the canvas. From the **Design** palette on the left, under the **General** folder, drag a **Start** job entry onto the canvas.



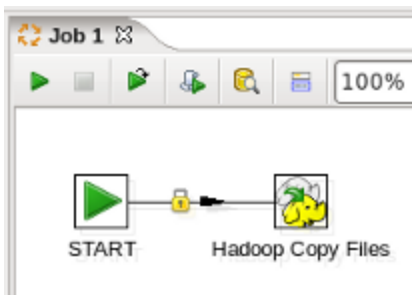
4. Add a Hadoop Copy Files job entry to the canvas. From the **Design** palette, under the **Big Data** folder, drag a **Hadoop Copy Files** job entry onto the canvas.



5. Connect the two job entries by hovering over the **Start** entry and selecting the output connector

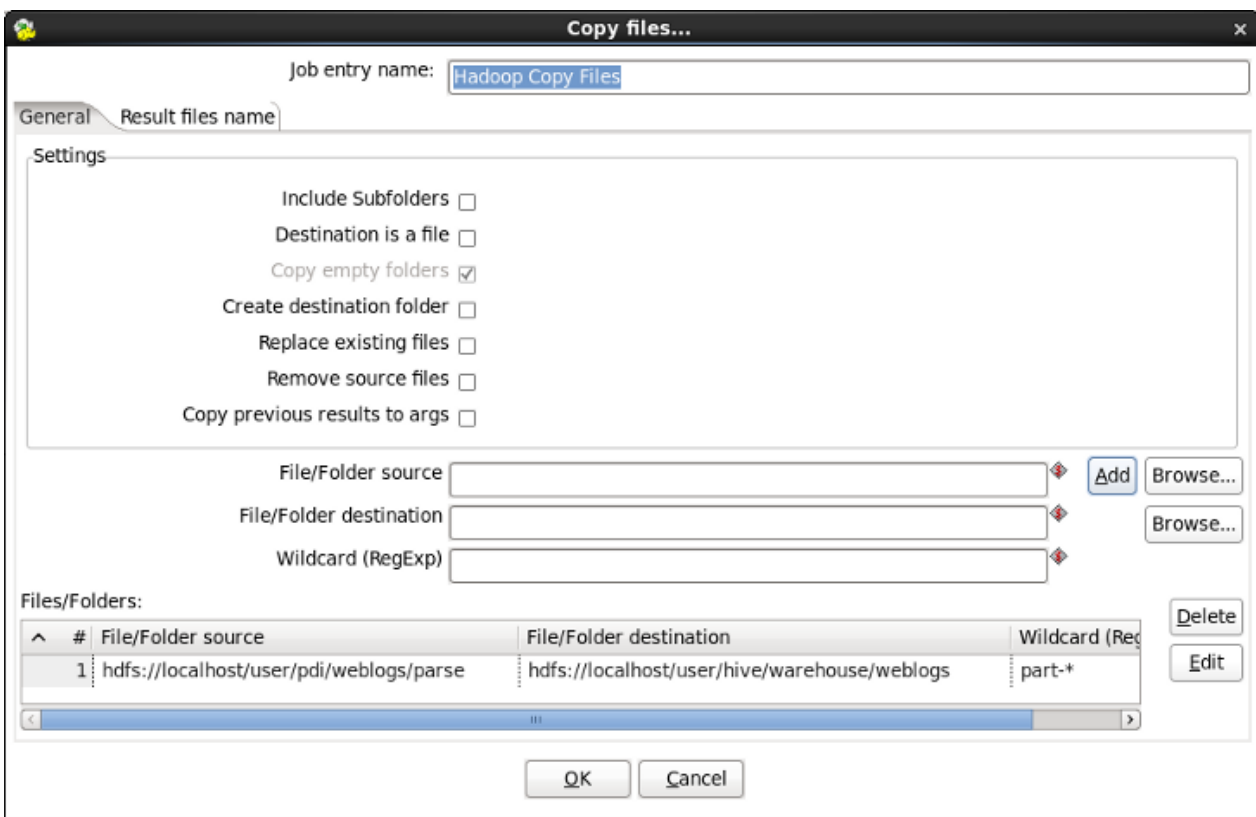


, then drag the connector arrow to the **Hadoop Copy Files** entry.



6. Enter the source and destination information within the properties of the **Hadoop Copy Files** entry by double-clicking it.
 - a. For **File/Folder source(s)**, enter `hdfs://<NAMENODE>:<PORT>/user/pdi/weblogs/parse`, where **NAMENODE** and **PORT** reflect your Hadoop destination.
 - b. For **File/Folder destination(s)**, enter `hdfs://<NAMENODE>:<PORT>/user/hive/warehouse/weblogs`.
 - c. For **Wildcard (RegExp)**, enter `part-.*`.
 - d. Click the **Add** button to add the entries to the list of files to copy.

When you are done your window should look like this (your file paths may be different)



Click **OK** to close the window.

7. Save the job by selecting **Save as** from the **File** menu. Enter `load_hive.kjb` as the file name within a folder of your choice.
8. Run the job by clicking the green Run button on the job toolbar



, or by selecting **Action > Run** from the menu. The **Execute a job** window opens. Click **Launch**.

An **Execution Results** panel opens at the bottom of the Spoon interface and displays the progress of the job as it runs. After a few seconds the job finishes successfully.

Job / Job Entry	Comment	Result	Reason	Filename
load_hive				
Job: load_hive	Start of job execution		start	
START	Start of job execution		start	
START	Job execution finished	Success		
Copy Files	Start of job execution		Followed unconditional link	
Copy Files	Job execution finished	Success		
Job: load_hive	Job execution finished	Success	finished	

If any errors occurred the job entry that failed will be highlighted in red and you can use the **Logging** tab to view error messages.

9. Verify the data was loaded by querying Hive.
 - a. Open the Hive shell from the command line by entering `hive`.
 - b. Enter this query to verify the data was loaded correctly into Hive.

```
select * from weblogs limit 10;
```

Ten rows of data are returned.

Using a Transformation Step to Load Data into HBase

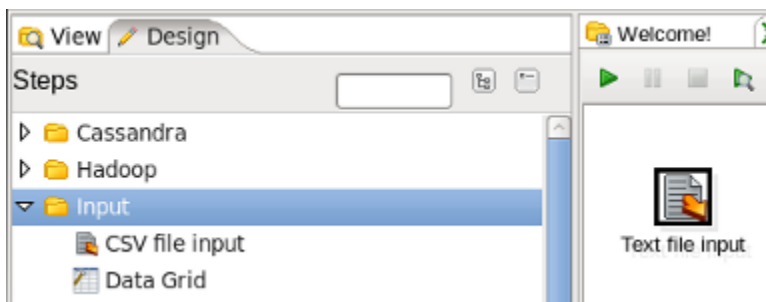
In order to follow along with this tutorial, you will need

- Hadoop
- Pentaho Data Integration
- HBase

This tutorial describes how to use data from a sample flat file to create a HBase table using a PDI transformation. For the sake of brevity, you will use a prepared sample dataset and a simple transformation to prepare and transform your data for HBase loads.

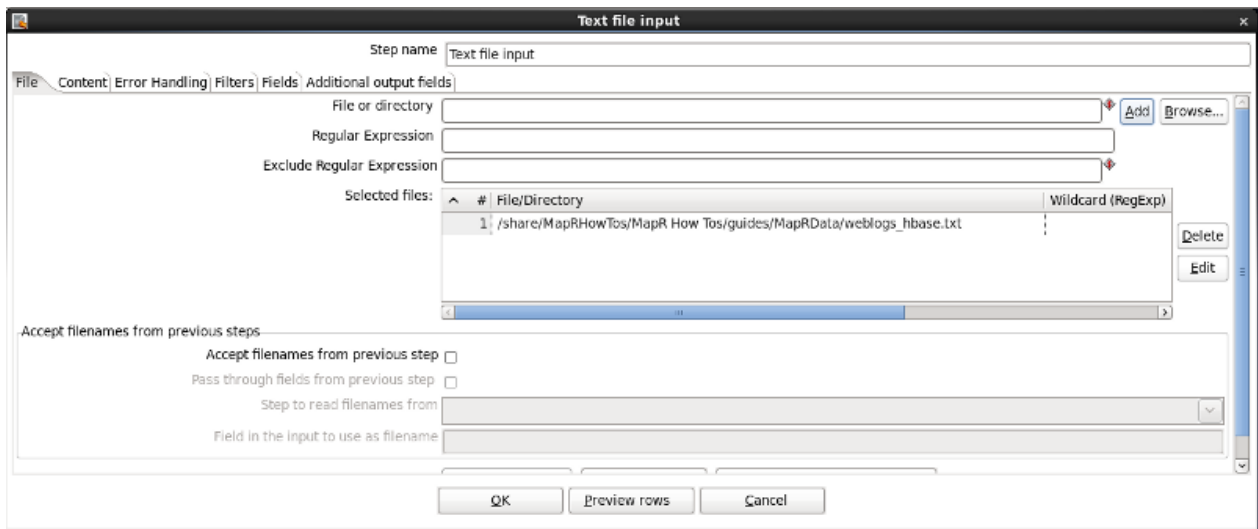
If not already running, start Hadoop, PDI, and HBase. Unzip the sample data files and put them in a convenient location: [weblogs_hbase.txt.zip](#)

1. Create a HBase Table.
 - a. Open the HBase shell by entering `hbase shell` at the command line.
 - b. Create the table in HBase by entering `create 'weblogs', 'pageviews'` in the HBase shell. This creates a table named `weblogs` with a single column family named `pageviews`.
 - c. Close the HBase shell by entering `quit`.
2. From within the Spoon, create a new transformation by selecting **File > New > Transformation**.
3. Identify the source where the transformation will get data from. For this tutorial your source is a text file (`.txt`). From the **Input** folder of the **Design** palette on the left, add a **Text File Input** step to the transformation by dragging it onto the canvas.

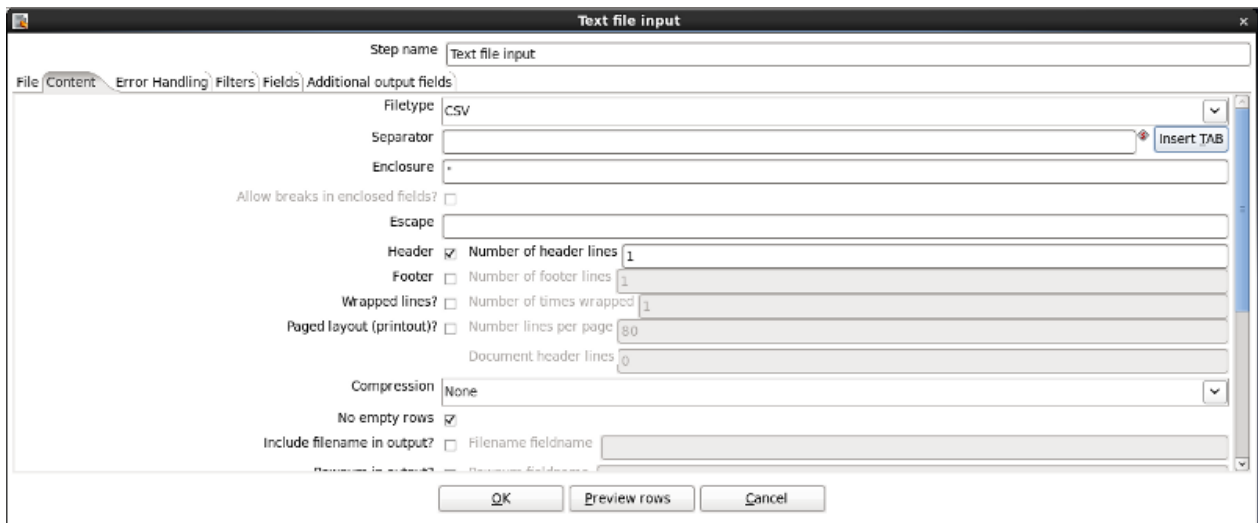


4. Edit the properties of the **Text file input** step by double-clicking the icon. The **Text file input** dialog box appears.
5. From the **File** tab, in the **File or Directory** field, click **Browse** and navigate to the `weblog_hbase.txt` file. Click **Add**.

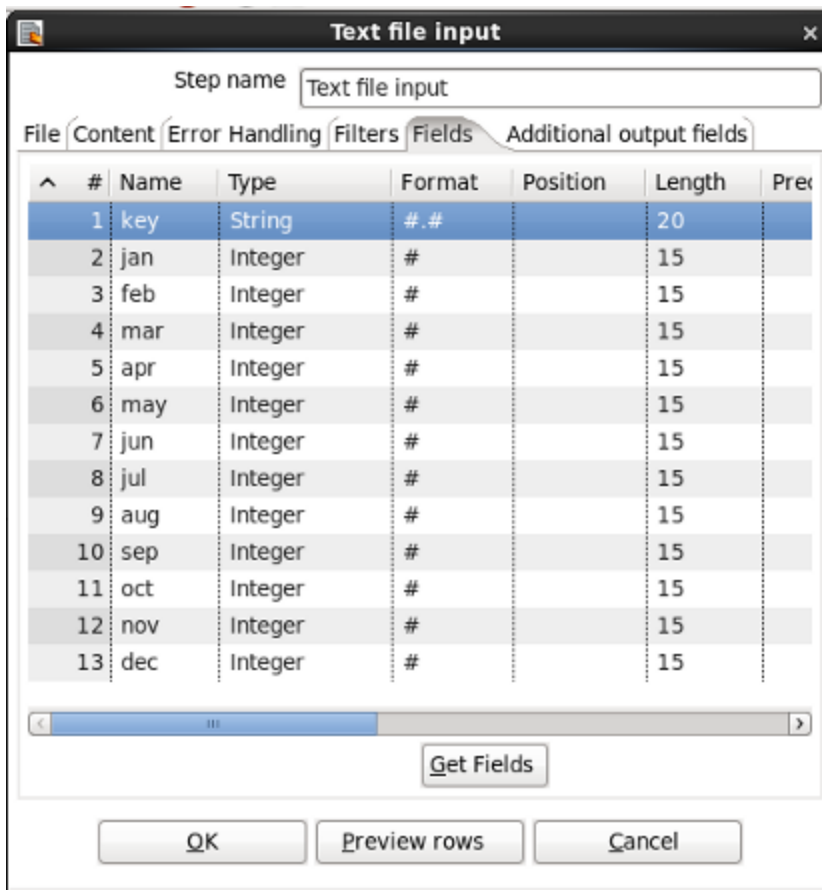
The file appears in the **Selected files** pane.



6. Configure the contents of the file by switching to the **Content** tab.
 - a. For **Separator**, clear the contents and click **Insert TAB**.
 - b. Check the **Header** checkbox.
 - c. For **Format**, Select **Unix** from the drop-down menu.

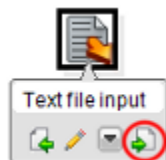


7. Configure the input fields.
 - a. From the **Fields** tab, select **Get Fields** to populate the list the available fields.
 - b. A dialog box appears asking for **Number of sample lines**. Enter **100** and click **OK**.
 - c. Change the **Type** of the field named **key** to **String** and set the **Length** to **20**.

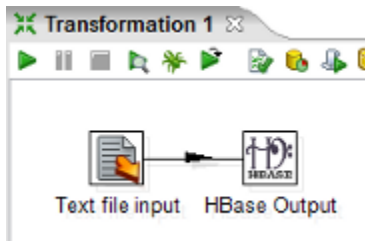


Click **OK** to close the window.

8. On the **Design** palette, under **Big Data**, drag the **HBase Output** to the canvas. Create a hop to connect your input and **HBase Output** step by hovering over the input step and clicking the output connector

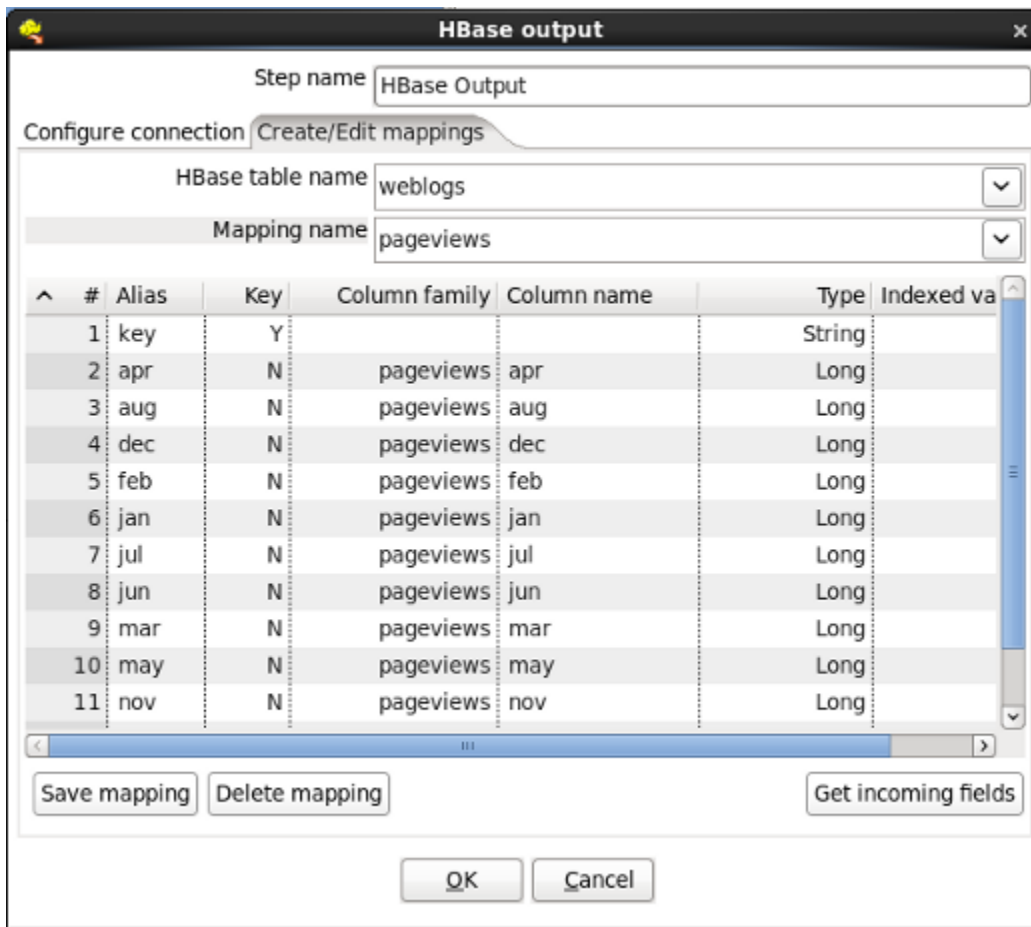


, then drag the connector arrow to the **HBase Output** step.



9. Edit the **HBase Output** step by double-clicking it. You must now enter your Zookeeper host(s) and port number.
 - a. For the **Zookeeper hosts(s)** field, enter a comma separated list of your HBase Zookeeper Hosts. For local single node clusters use `localhost`.
 - b. For **Zookeeper port**, enter the port for your Zookeeper hosts. By default this is `2181`.
10. Create a HBase mapping to tell Pentaho how to store the data in HBase by switching to the **Create/Edit mappings** tab and changing these options.

- For **HBase table name**, select **weblogs**.
- For **Mapping name**, enter **pageviews**.
- Click **Get incoming fields**.
- For the alias **key** change the **Key** column to **Y**, clear the **Column family** and **Column name** fields, and set the **Type** field to **String**. Click **Save mapping**.



- Configure the HBase out to use the mapping you just created.
 - Go back to the **Configure connection** tab and click **Get table names**.
 - For **HBase table name**, enter **weblogs**.
 - Click **Get mappings for the specified table**.
 - For **Mapping name**, select **pageviews**. Click **OK** to close the window.

Save the transformation by selecting **Save as** from the **File** menu. Enter **load_hbase.ktr** as the file name within a folder of your choice.

- Run the transformation by clicking the green **Run** button on the transformation toolbar



, or by choosing **Action > Run** from the menu. The **Execute a transformation** window opens. Click **Launch**.

An **Execution Results** panel opens at the bottom of the Spoon interface and displays the progress of the transformation as it runs. After a few seconds the transformation finishes successfully.

Execution Results

Execution History

Logging

Step Metrics

Performance Graph

^	#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active
	1	Text file input	0	0	27300	27301	0	1	0	0	Finished
	2	HBase Output	0	27300	27300	0	0	0	0	0	Finished

If any errors occurred the transformation step that failed will be highlighted in red and you can use the **Logging** tab to view error messages.

13. Verify the data was loaded by querying HBase.
 - a. From the command line, open the HBase shell by entering this command.

```
hbase shell
```

- b. Query HBase by entering this command.

```
scan 'weblogs', {LIMIT => 10}
```

Ten rows of data are returned.

Transforming Data within a Hadoop Cluster

These tutorials contain guidance and instructions on transforming data within the Hadoop cluster using Pentaho MapReduce, Hive, and Pig.

- [Using Pentaho MapReduce to Parse Weblog Data](#)—How to use Pentaho MapReduce to convert raw weblog data into parsed, delimited records.
- [Using Pentaho MapReduce to Generate an Aggregate Dataset](#)—How to use Pentaho MapReduce to transform and summarize detailed data into an aggregate dataset.
- [Transforming Data within Hive](#)—How to read data from a Hive table, transform it, and write it to a Hive table within the workflow of a PDI job.
- [Transforming Data with Pig](#)—How to invoke a Pig script from a PDI job.

Extracting Data from a Hadoop Cluster

These tutorials contain guidance and instructions on extracting data from Hadoop using HDFS, Hive, and HBase.

- [Extracting Data from HDFS to Load an RDBMS](#)—How to use a PDI transformation to extract data from HDFS and load it into a RDBMS table.
- [Extracting Data from Hive to Load an RDBMS](#)—How to use a PDI transformation to extract data from Hive and load it into a RDBMS table.
- [Extracting Data from HBase to Load an RDBMS](#)—How to use a PDI transformation to extract data from HBase and load it into a RDBMS table.
- [Extracting Data from Snappy Compressed Files](#)—How to configure client-side PDI so that files compressed using the Snappy codec can be decompressed using the Hadoop file input or Text file input step.

Reporting on Data within a Hadoop Cluster

These tutorials contain guidance and instructions about reporting on data within a Hadoop cluster.

- [Reporting on HDFS File Data](#)—How to create a report that sources data from a HDFS file.
- [Reporting on HBase Data](#)—How to create a report that sources data from HBase.
- [Reporting on Hive Data](#)—How to create a report that sources data from Hive.

MapR Tutorials

These tutorials are organized by topic and each set explains various techniques for loading, transforming, extracting and reporting on data within a MapR cluster. You are encouraged to perform the tutorials in order as the output of one is sometimes used as the input of another. However, if you would like to jump to a tutorial in the middle of the flow, instructions for preparing input data are provided.

- [Loading Data into a MapR Cluster](#)
- [Transforming Data within a MapR Cluster](#)
- [Extracting Data from a MapR Cluster](#)
- [Reporting on Data within a MapR Cluster](#)

Loading Data into a MapR Cluster

These tutorials contain guidance and instructions on loading data into CLDB (MapR's distributed file system), Hive, and HBase.

- [Loading Data into CLDB](#)—How to use a PDI job to move a file into CLDB.
- [Loading Data into MapR Hive](#)—How to use a PDI job to load a data file into a Hive table.
- [Loading Data into MapR HBase](#)—How to use a PDI transformation that sources data from a flat file and writes to an HBase table.

Transforming Data within a MapR Cluster

These tutorials contain guidance and instructions on leveraging the massively parallel, fault tolerant MapR processing engine to transform resident cluster data.

- [Using Pentaho MapReduce to Parse Weblog Data in MapR](#)—How to use Pentaho MapReduce to convert raw weblog data into parsed, delimited records.
- [Using Pentaho MapReduce to Generate an Aggregate Dataset in MapR](#)—How to use Pentaho MapReduce to transform and summarize detailed data into an aggregate dataset.
- [Transforming Data within Hive in MapR](#)—How to read data from a Hive table, transform it, and write it to a Hive table within the workflow of a PDI job.
- [Transforming Data with Pig in MapR](#)—How to invoke a Pig script from a PDI job.

Extracting Data from a MapR Cluster

These tutorials contain guidance and instructions on extracting data from a MapR cluster and loading it into an RDBMS table.

- [Extracting Data from CLDB to Load an RDBMS](#)—How to use a PDI transformation to extract data from MapR CLDB and load it into a RDBMS table.
- [Extracting Data from Hive to Load an RDBMS in MapR](#)—How to use a PDI transformation to extract data from Hive and load it into a RDBMS table.
- [Extracting Data from HBase to Load an RDBMS in MapR](#)—How to use a PDI transformation to extract data from HBase and load it into a RDBMS table.

Reporting on Data within a MapR Cluster

These tutorials contain guidance and instructions about reporting on data within a MapR cluster.

- [Reporting on CLDB File Data](#) —How to create a report that sources data from a MapR CLDB file.
- [Reporting on HBase Data in MapR](#)—How to create a report that sources data from HBase.
- [Reporting on Hive Data in MapR](#)—How to create a report that sources data from Hive.

Cassandra Tutorials

These tutorials demonstrate the integration between Pentaho and the Cassandra NoSQL Database, specifically techniques about writing data to and reading data from Cassandra using graphical tools. These tutorials also include instructions on how to sort and group data, create reports, and combine data from Cassandra with data from other sources.

- [Write Data To Cassandra](#)—How to read data from a data source (flat file) and write it to a column family in Cassandra using a graphic tool.
- [How To Read Data From Cassandra](#)—How to read data from a column family in Cassandra using a graphic tool.
- [How To Create a Report with Cassandra](#)—How to create a report that uses data from a column family in Cassandra using graphic tools.

MongoDB Tutorials

These tutorials demonstrate the integration between Pentaho and the MongoDB NoSQL Database, specifically how to write data to, read data from, MongoDB using graphical tools. These tutorials also include instructions on sorting and grouping data, creating reports, and combining data from Mongo with data from other sources.

- [Write Data To MongoDB](#)—How to read data from a data source (flat file) and write it to a collection in MongoDB
- [Read Data From MongoDB](#)—How to read data from a collection in MongoDB.
- [Create a Report with MongoDB](#)—How to create a report that uses data from a collection in MongoDB.
- [Create a Parameterized Report with MongoDB](#)—How to create a parameterize report that uses data from a collection in MongoDB.

PDI Hadoop Configurations

Within PDI, a Hadoop configuration is the collection of Hadoop libraries required to communicate with a specific version of Hadoop and related tools, such as Hive HBase, Sqoop, or Pig.

Hadoop configurations are defined in the `plugin.properties` file and are designed to be easily configured within PDI by changing the `active.hadoop.configuration` property. The `plugin.properties` file resides in the `pentaho-big-data-plugin/` folder.

All Hadoop configurations share a basic structure. Elements of the structure are defined in the table following this code block.

```
configuration/  
|-- lib/  
|-- |-- client/  
|-- |-- pmr/  
|-- '--- *.jar  
|-- config.properties  
|-- core-site.xml  
`-- configuration-implementation.jar
```

Configuration Element	Definition
lib/	Libraries specific to the version of Hadoop this configuration was created to communicate with.
client/	Libraries that are only required on a Hadoop client, for instance <code>hadoop-core-*</code> or <code>hadoop-client-*</code>
pmr/	Jar files that contain libraries required for parsing data in input/output formats or otherwise outside of any PDI-based execution.
*.jar	All other libraries required for Hadoop configuration that are not client-only or special <code>pmr</code> jar files that need to be available to the entire JVM of Hadoop job tasks.
config.properties	Contains metadata and configuration options for this Hadoop configuration. Provides a way to define a configuration name, additional classpath, and

	native libraries the configuration requires. See the comments in this file for more details.
core-site.xml	Configuration file that can be replaced to set a site-specific configuration, for example <code>hdfs-site.xml</code> would be used to configure HDFS.
configuration-implementation.jar	File that must be replaced in order to communicate with this configuration.

- [Create a New Hadoop Configuration](#)
- [Include or Exclude Classes or Packages for a Hadoop Configuration](#)

Create a New Hadoop Configuration

If you have a Hadoop distribution not supported by Pentaho, or you have modified your Hadoop Installation in such a way that it is no longer compatible with Pentaho, you may need to create a new Hadoop configuration.

Changing which version of Hadoop PDI can communicate with requires you to swap the appropriate `jar` files within the plugin directory and then update the `plugin.properties` file.

CAUTION:

Creating a new Hadoop configuration is not officially supported by Pentaho. Please inform Pentaho support regarding your requirements.

1. Identify which Hadoop configuration most closely matches the version of Hadoop you want to communicate with. If you compare the default configurations included the differences are apparent. Copy this folder, then paste and rename it. The name of this folder will be the name of your new configuration.
2. Copy the `jar` files for your specified Hadoop version.
3. Paste the `jar` files into the `lib/` directory.
4. Change the `active.hadoop.configuration=` property in the `plugins/pentaho-big-dataplugin/plugin.properties` file to match your specific Hadoop configuration. This property configures which distribution of Hadoop to use when communicating with a Hadoop cluster and must match the name of the folder you created in Step 1. Update this property if you are using a version other than the default Hadoop version.

Include or Exclude Classes or Packages for a Hadoop Configuration

You have the option to include or exclude classes or packages from loading with a Hadoop configuration. Configure these options within the `plugin.properties` file located at `plugins/pentaho-big-data-plugin`. For additional information, see the comments within the `plugin.properties` file.

Including Additional Class Paths or Libraries

To include additional class paths, native libraries, or a user-friendly configuration name, include the directory within `classpath` property within the big data `plugin.properties` file.

Exclude Classes or Packages

To exclude classes or packages from being loaded twice by a Hadoop configuration class loader, include them in the `ignored.classes` property within the `plugin.properties` file. This is necessary when logging libraries expect a single class shared by all class loaders, as with Apache Commons Logging for example.

PDI Big Data Transformation Steps

This section contains reference documentation for transformation steps which enable PDI to work with big data technologies.

Please see [Create DI Solutions](#) for additional transformation step references.

- [Avro Input](#)
- [Cassandra Input](#)
- [Cassandra Output](#)
- [CouchDB](#)
- [Hadoop File Input](#)
- [Hadoop File Output](#)
- [HBase Input](#)
- [HBase Output](#)
- [HBase Row Decoder](#)
- [MapReduce Input](#)
- [MapReduce Output](#)
- [MongoDB Input](#)
- [MongoDB Output](#)
- [Splunk Input](#)
- [Splunk Output](#)
- [SSTable Output](#)

PDI Big Data Job Entries

This section contains reference documentation for job entries which enable PDI to work with big data technologies.

Please see [Using Pentaho Data Integration](#) for additional transformation step and job entry references.

- [Amazon EMR Job Executor](#)
- [Amazon Hive Job Executor](#)
- [Hadoop Copy Files](#)
- [Hadoop Job Executor](#)
- [Oozie Job Executor](#)
- [Pentaho MapReduce](#)
- [Pig Script Executor](#)
- [Sqoop Export](#)
- [Sqoop Import](#)
- [Start a YARN Kettle Cluster](#)
- [Stop a YARN Kettle Cluster](#)