Contents

Performance Monitoring and Logging

Performance Monitoring and Logging

Monitoring Step Performance
- Using Performance Graphs

Logging Steps

Logging Transformations

Pentaho Data Integration Performance Tuning Tips

Performance Monitoring and Logging

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

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

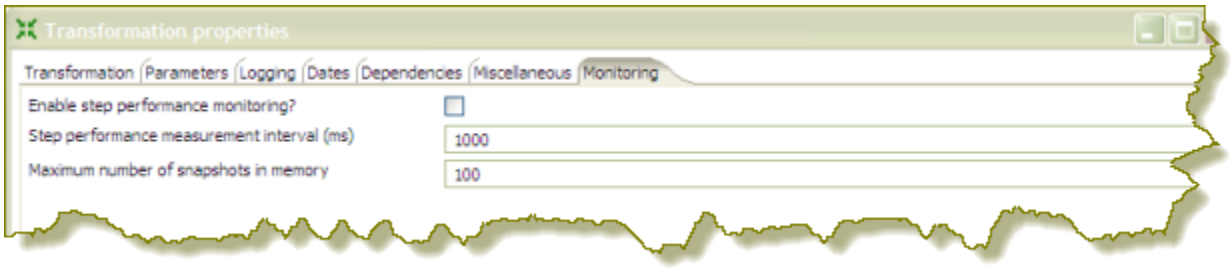
If you are an administrative user and want to monitor jobs and transformations, you must first set up logging and performance monitoring in Spoon. For more information about monitoring jobs and transformations, see the section Administer the DI Server.

- Monitoring Step Performance
- Logging Steps
- Logging Transformations
- Pentaho Data Integration Performance Tuning Tips

Monitoring Step Performance

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

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

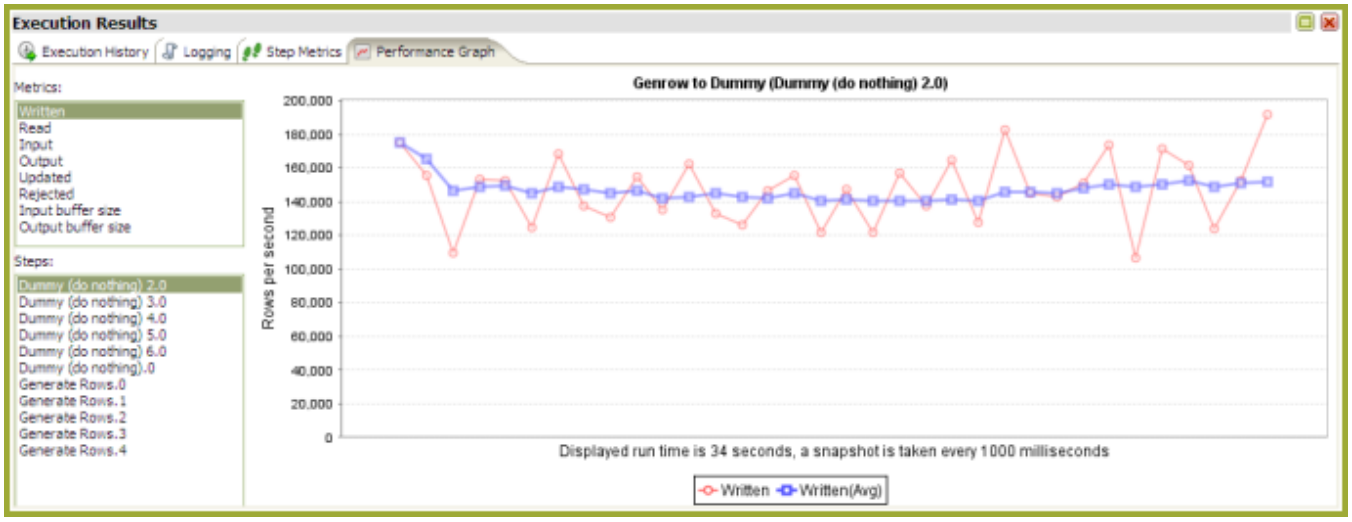


As shown in the sample screen capture above, the option to track performance (**Enable step performance monitoring?**) is not selected by default. Step performance monitoring may cause memory consumption problems in long-running transformations. By default, a performance snapshot is taken for all the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

- Using Performance Graphs

Using Performance Graphs

If you configured step performance monitoring, with database logging (optional), you can view the performance evolution graphs. Performance graphs provide you with a visual interpretation of how your transformation is processing. To enable database logging, enable the option **Enable step performance monitoring** within the **Transformation Properties / Monitoring** dialog box.



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

1. Right-click in the workspace (canvas) where you have an open transformation. Alternatively, press <CTRL +T>. To enable the logging, you also need to enable the option **Enable step performance monitoring in the Transformation Properties/Monitoring** in the dialog. The **Transformation Properties** dialog box appears.
2. In the Transformation Properties dialog box, click the **Logging** tab. Make sure **Performance** is selected in the navigation pane on the left.
3. Under **Logging** enter the following information:

Option	Description
Log Connection	Specifies the database connection you are using for logging; you can configure a new connection by clicking **New**.
Log Table Schema	Specifies the schema name, if supported by your database
Log Table Name	Specifies the name of the log table (for example L_ETL)
Logging interval (seconds)	Specifies the interval in which logs are written to the table

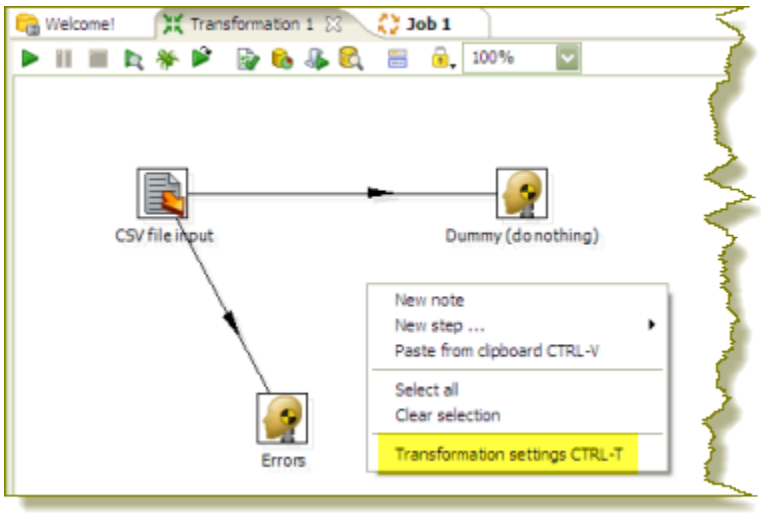
Option	Description
Log record timeout (in days)	Specifies the number of days old log entries in the table will be kept before they are deleted

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

Logging Steps

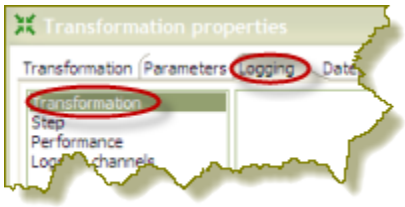
Follow the instructions below to create a log table that keeps history of step-related information associated with your transformation.

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



   The **Transformation Properties** dialog box appears.

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



3. Under **Logging** enter the following information:

Option	Description
Log Connection	Specifies the database connection you are using for logging; you can configure a new connection by clicking **New**.
Log Table Schema	Specifies the schema name, if supported by your database
Log Table Name	Specifies the name of the log table (for example L_STEP)

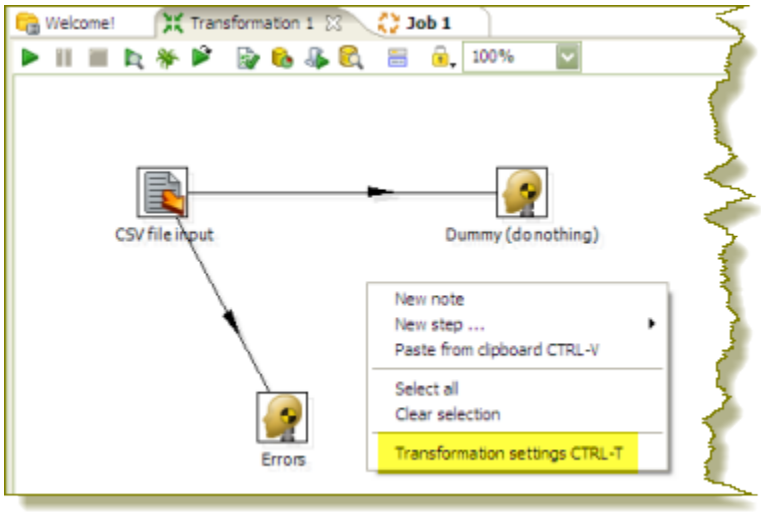
Option	Description
Logging interval (seconds)	Specifies the interval in which logs are written to the table
Log record timeout (in days)	Specifies the number of days old log entries in the table will be kept before they are deleted

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

Logging Transformations

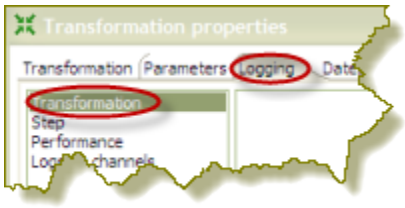
Follow the instructions below to create a log table for transformation-related processes:

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



   The **Transformation Properties** dialog box appears.

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



3. Under **Logging** enter the following information:

Option	Description
Log Connection	Specifies the database connection you are using for logging; you can configure a new connection by clicking **New**.
Log Table Schema	Specifies the schema name, if supported by your database
Log Table Name	Specifies the name of the log table (for example L_ETL)

Option	Description
Logging interval (seconds)	Specifies the interval in which logs are written to the table
Log record timeout (in days)	Specifies the number of days old log entries in the table will be kept before they are deleted
Log size limit in lines	Limits the number of lines that are stored in the LOG_FIELD (when selected under Fields to Log); when the LOG_FIELD is enabled Pentaho Data Integration will store logging associated with the transformation in a long text field (CLOB)

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

The next time you run your transformation, logging information will be displayed under the **Execution History** tab.

Pentaho Data Integration Performance Tuning Tips

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

Step	Tip	Description
JS	Turn off compatibility mode	Rewriting JavaScript to use a format that is not compatible with previous versions is, in most instances, easy to do and makes scripts easier to work with and to read. By default, old JavaScript programs run in compatibility mode. That means that the step will process like it did in a previous version. You may see a small performance drop because of the overload associated with forcing compatibility. If you want make use of the new architecture, disable compatibility mode and change the code as shown below: • `intField.getInteger() > intField` • `numberField.getNumber() > numberField` • `dateField.getDate() > dateField` • `bigNumberField.getBigNumber() > bigNumberField` • and so on... Instead of Java methods, use the built-in library. Notice that the resulting program code is more intuitive. For example : • checking for null is now: `field.isNull() > field==null` • Converting string to date: `field.Clone().str2dat() > str2date(field)` • and so on... If you convert your code as shown above, you may get significant performance benefits. Note: It is no longer possible to modify data in-place using the value methods. This was a design decision to ensure that no data with the wrong type would end up in the output rows of the step. Instead of modifying fields in-place, create new fields using the table at the bottom of the Modified JavaScript transformation.
JS	Combine steps	One large JavaScript step runs faster than three consecutive smaller steps. Combining processes in one larger step helps to reduce overhead.
JS	Avoid the JavaScript step or write a custom plug in	Remember that while JavaScript is the fastest scripting language for Java, it is still a scripting language. If you do the same amount of work in a native step or plugin, you avoid the overhead of the JS scripting engine. This has been known to result in significant performance gains. It is also the primary reason why the Calculator step was created — to avoid the use of JavaScript for simple calculations.
JS	Create a copy of a field	No JavaScript is required for this; a "Select Values" step does the trick. You can specify the same field twice. Once without a rename, once (or more) with a rename. Another trick is to use B=NVL(A,A) in a Calculator step where B is forced

Step	Tip	Description
		to be a copy of A. In version 3.1, an explicit "create copy of field A" function was added to the Calculator.
JS	Data conversion	Consider performing conversions between data types (dates, numeric data, and so on) in a "Select Values" step (version 3.0.2 or higher). You can do this in the Metadata tab of the step.
JS	Variable creation	If you have variables that can be declared once at the beginning of the transformation, make sure you put them in a separate script and mark that script as a startup script (right click on the script name in the tab). JavaScript object creation is time consuming so if you can avoid creating a new object for every row you are transforming, this will translate to a performance boost for the step.
N/A	Launch several copies of a step	There are two important reasons why launching multiple copies of a step may result in better performance: 1. The step uses a lot of CPU resources and you have multiple processor cores in your computer. Example: a JavaScript step 2. Network latencies and launching multiple copies of a step can reduce average latency. If you have a low network latency of say 5ms and you need to do a round trip to the database, the maximum performance you get is 200 (x5) rows per second, even if the database is running smoothly. You can try to reduce the round trips with caching, but if not, you can try to run multiple copies. Example: a database lookup or table output
N/A	Manage thread priorities	In versions 3.0.2 and higher, this feature that is found in the "Transformation Settings" dialog box under the (Misc tab) improves performance by reducing the locking overhead in certain situations. This feature is enabled by default for new transformations that are created in recent versions, but for older transformations this can be different.
Select Value	If possible, don't remove fields in Select Value	Don't remove fields in Select Value unless you must. It's a CPU-intensive task as the engine needs to reconstruct the complete row. It is almost always faster to add fields to a row rather than delete fields from a row.
Get Variables	Watch your use of Get Variables	May cause bottlenecks if you use it in a high-volume stream (accepting input). To solve the problem, take the "Get Variables" step out of the transformation (right click, detach)then insert it in with a "Join Rows (cart prod)" step. Make sure to specify the main step from which to read in the "Join Rows" step. Set it to the step that originally provided the "Get Variables" step with data.
N/A	Use new text file input	The new "CSV Input" or "Fixed Input" steps provide optimal performance. If you have a fixed width (field/row) input file, you can even read data in parallel. (multiple copies) These new steps have been rewritten using Non-blocking I/O (NIO) features. Typically, the larger the NIO buffer you specify in the step, the better your read performance will be.
N/A	When appropriate, use lazy conversion	In instances in which you are reading data from a text file and you write the data back to a text file, use Lazy conversion to speed up the process. The principle behind lazy conversion that it delays data conversion in hopes that it isn't necessary (reading from a file and writing it back comes to mind). Beyond helping with data conversion, lazy conversion also helps to keep the data in "binary" storage form. This, in turn, helps the internal Kettle engine to perform faster data serialization (sort, clustering, and so on). The Lazy Conversion option is available in the "CSV Input" and "Fixed input" text file reading steps.

Step	Tip	Description
Join Rows	Use Join Rows	You need to specify the main step from which to read. This prevents the step from performing any unnecessary spooling to disk. If you are joining with a set of data that can fit into memory, make sure that the cache size (in rows of data) is large enough. This prevents (slow) spooling to disk.
N/A	Review the big picture: database, commit size, row set size and other factors	Consider how the whole environment influences performance. There can be limiting factors in the transformation itself and limiting factors that result from other applications and PDI. Performance depends on your database, your tables, indexes, the JDBC driver, your hardware, speed of the LAN connection to the database, the row size of data and your transformation itself. Test performance using different commit sizes and changing the number of rows in row sets in your transformation settings. Change buffer sizes in your JDBC drivers or database.
N/A	Step Performance Monitoring	Step Performance Monitoring is an important tool that allows you identify the slowest step in your transformation.