Best Practices - PDI Logging & Monitoring

This page intentionally left blank.

Contents

This page intentionally left blank.

Overview

The main objective of this document is to provide information about different options for best practices associated with PDI logging. Some of the things discussed here include why you should use PDI logging, levels of logging, transformation and job logging, and debugging transformations and jobs.

The intention of this document is to speak about topics generally; however, these are the specific versions covered here:

Software	Version
Pentaho	5.4, 6.x, 7.x

Here are some terms you should know before we begin:

- Logging – refers to file logging.
- Monitoring – examines the impact of operational needs on network performance.

Pentaho Data Integration (PDI) Logging

Every process executed within PDI will have output information related to workflow logging. This gives you details about what is happening during execution. Logs can be monitored through the PDI client or the Kettle online interface.

Here is a list of things that logging can help with:

- Provides relevant information whenever a process execution has an error, such as which steps are failing and trace with main error description.
- Gives information about a workflow if it has decision split.
- Detects bottlenecks and bad performance steps based on a procedure's duration; for example, stored execution times can be used to detect if a process is taking longer than usual.
- Shows status of currently running processes. Logs provide information about when the process started, where it is right now, and data related to its status.
- Traceability of what has been done, and when.

Levels of Logging

PDI offers the possibility to establish different levels of logging verbosity depending on your needs:

Table 1: Logging Levels

Logging Levels	Definitions
Nothing	Logging is enabled but does not record any output.
Error	Only shows error lines.
Minimal	Only uses minimal logging. Gives information regarding the workflows status.
Basic	**Recommendation:** Use the Basic (default) logging level. It shows information related to every step.
Detailed	**Use for Troubleshooting:** gives detailed login output.
Debug	A detailed output for debugging purposes. The debug logging level should never be used in a production environment.
Row Level	Logging at a row-level detail. This generates a huge amount of log output data.

Best Practices for Logging Levels

Logging levels should be lower in a production or QA environment, but can be higher in a development or non-production environment.

*The **Debug** logging level should never be used in a production environment.*

Logging levels can also be specified when the process is executed with Spoon or any command line tool. Enable Logging in Pentaho Documentation has information about using PDI for logging.

Use Low Level Logging in Production or QA or Use Higher Logging Levels for Debugging

- **Recommendation:** Use the **lowest** logging level possible when working in **production** or **QA** environments.
- **Recommendation:** Use **higher** logging levels for specific process **debugging but never in a production environment**.
- **Rationale:** Your process performance may be affected if the logging level is **detailed**. This will also increase the amount of information stored in the log.

Log Rotation Types

Logging functionality in Data Integration (DI) enables you to easily troubleshoot complex errors and failures, and measure performance. There is no preferred way to store execution logs, but it is usually recommended that you follow enterprise best practices to define which method is good for you. Pentaho processes and stores logging within log files into a filesystem. These files can be separated into two types: server log files and single process execution log files.

DI Server Log Files

The DI server stores execution log files in a single file, located in the DI server path within the logs folder. This file has execution details from jobs and transformations within the DI server.

We recommend establishing a log rotation pattern in production environments. This provides an easy way to locate single execution information, avoid huge file creation, and establish maintenance policies for old log files.

The following recommended patterns are common and useful:

- **Daily log rotation**: Log files are stored in separate files depending on the OS date with compression.
- **Size-based log rotation**: Log files are stored in separate files when there is a rich amount of space permitted per file.

If you are using Carte for logging, be aware that Carte logs can get large.

Visit Pentaho's help page regarding log rotation for more information about specifying rotation patterns.

Single Process Execution Log File

PDI's command line tools are typically used to execute PDI content, such as transformations and jobs, from outside Spoon. These commands are `Pan` and `Kitchen`. They generate logs related to process execution results.

Visit Pentaho's help page regarding the command line tools Pan and Kitchen for more information about specifying log destination.

Best Practices for Log Files

The following are best practices for using log files in PDI to ensure quality performance.

Use a Single File for Root Jobs/Transformations

- **Recommendation:** Use a sub job that writes to one log file for the entire execution. This approach is recommended for processes executed in the DI Server and Carte.
- **Rationale:** This organizes all logging related to that job into one file, which facilitates research when evaluating an error in a job if that server runs more than one job at a time.

Use Centralized Location for Logs

- **Recommendation:** Place logs in a central location to make log information easier to find. This could be the common Project Directory logs directory. The PDI Development best practices article has more information about PDI directories.
- **Rationale:** Centralized logging allows you to take advantage of PDI's performance.

Make Sure File Names and Paths Are Consistent with Process Names

- **Recommendation:** Log file destination and levels must be specified when the command line is invoked for proper use. The timestamp of this execution should distinguish it from other executions and processes.
- **Rationale:** The file name and path should be consistent with the process name to be executed.

Transformations and Job Logging

Transformations and jobs have logging capabilities, but the information obtained in each is different in nature. Jobs show information related to entry execution, such as when an entry is started and finished. Transformations show information about start and finish time, as well as steps executed and the number of rows processed.

Each job entry and transformation writes log information concerning their own processing. This information can be as detailed as needed depending on the logging levels used.

General recommendations:

- Use step names to search for log entries within files, databases or SNMP traps.

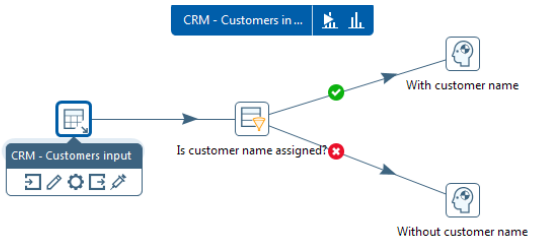

Figure 1: Naming Steps Example

  This allows you to easily diagnose your process and follow it execution. i.e. For "table input" steps, indicate connection name and input data description, with "Filter Rows" step, describe condition used as part of the name. Even with "dummy" steps, is important to specify a description of the stream expected
- Modify logging level to get more information out of the execution.

Debugging KTR and KJB Tips

Developers must be able to debug processes and find the information needed regarding execution behavior. The following recommendations are arranged in three different scenarios: execution errors, weird behavior, and performance issues.

Error. Transformation is throwing execution error

1. Identify which step is failing in the transformation.
2. Establish a higher level of detail in the log to get better error description and details. A major level of detail can give more information of the error.
3. Disable downstream of the transformation. Discard other steps to isolate the problem.
4. Observe if the error is due to a data issue or bad configuration of the step. Check the error messages for clues regarding the nature of the error.
5. Note if the error happens with a specific row in the stream or as soon the first record arrives. This might mean something is wrong or different with part of the data, rather than the step configuration itself.

Analyze the input stream of the step. Preview the transformation with or without conditions to observe data flow.

Unexpected behavior. Transformation is not behaving as expected

1. Preview the transformation to find which step is causing the problem
2. Disable downstream steps. Isolate part of the transformation to make the resolution easier and reduce execution times.
3. Use specific log steps to show specific variable and field values.
4. Incorporate other steps like **Filter rows** to understand if conditions are being performed as expected.

Troubleshooting Performance Issues

1. Identify which step(s) are causing bottlenecks; this can be indicated by an output speed that is slower than input speed. Try to identify if the issue is due to external resources, ETL design, or step slowness. These last are easy to identify in Spoon, as slow steps are represented with a dotted square around the step during the execution.
2. Replace input and output sources to discard 3rd party issues (e.g. network latency, database issues, etc.). Instead use steps like **Generate Rows** or serialize/de-serialize steps
3. Execute part of the transformation and study read rows vs write rows statistics showed in the execution panel tab.

Monitoring

PDI gives you the ability to monitor ETL processes; this includes DI events, and DI server or PDI process executions.

SNMP Traps

The Simple Network Management Protocol (SNMP) plugin gives large enterprises the ability to integrate with 3rd-party tools to monitor DI events. The SNMP allows you to monitor the DI server or PDI process execution.

Make sure to keep in mind that this output option is only available since Pentaho 6.0 and is recommended only if a 3rd party enterprise monitoring tool already exists.

Read Monitoring with PDI and SNMP Traps for more information on monitoring DI events and PDI process execution.

Related Information

The following links provide useful information related to topics discussed in this document:

- Best Practices – PDI Development
- Performance Monitoring and Logging
- Pan Options and Syntax
- Kitchen Options and Syntax
- Enable Logging
- Monitoring with PDI and SNMP Traps