

**PDI Techniques –  
Logging and Monitoring**

# HITACHI

Inspire the Next

Change log (if you want to use it):

Date	Version	Author	Changes

# Contents

- Overview..... 1
  - Before You Begin..... 1
    - Terms You Should Know..... 1
    - Use Case: Setting Appropriate Logging Levels..... 1
- Pentaho Data Integration (PDI) Logging..... 2
  - Levels of Logging..... 2
    - Best Practices for Logging Levels..... 3
    - Logging Levels for Production, QA, and Debugging..... 3
  - Log Rotation Types..... 3
    - DI Server Log Files..... 3
    - Single Process Execution Log File ..... 4
    - Best Practices for Log Files ..... 4
    - Transformations and Job Logging ..... 4
    - Debugging Tips for Transformations and Jobs..... 5
- Monitoring..... 7
  - SNMP Traps..... 7
- Related Information..... 8
- Finalization Checklist..... 8

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 reasons you should use PDI logging, levels of logging, transformation and job logging, and debugging transformations and jobs.

Our intended audience consists of developers and any others who wish to use PDI logging for correcting process execution errors, detecting bottlenecks and substandard performance steps, and keeping track of progress.

Software	Version(s)
Pentaho	6.x, 7.x, 8.0

The [Components Reference](#) in Pentaho Documentation has a complete list of supported software and hardware.

## Before You Begin

This document assumes that you have knowledge about PDI logging and monitoring, and are familiar with environments that may require it for optimal performance.

### *Terms You Should Know*

Here are some terms you should be familiar with:

- [Logging](#): refers to file logging
- [Monitoring](#): examines the impact of operational needs on network performance

### *Use Case: Setting Appropriate Logging Levels*

---

*Fabiola is working in a QA environment and needs an appropriate logging level for the job. She sometimes works in a production environment, and has learned that the two environments require different levels of logging. If she uses the lowest level possible, she could accommodate both environments. However, if she is performing specific process debugging in either environment, she can use higher levels for her QA environment only.*

---

# 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 PDI 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 substandard 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 the status of currently running processes. Logs provide information about when the process started, where it is currently, 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
<b>Nothing</b>	Logging is enabled but does not record any output.
<b>Error</b>	Only shows error lines.
<b>Minimal</b>	Only uses minimal logging. Gives information regarding the workflows status.
<b>Basic</b>	<b>Recommendation:</b> Use the Basic (default) logging level. It shows information related to every step.
<b>Detailed</b>	<b>Use for Troubleshooting:</b> gives detailed login output.
<b>Debug</b>	A detailed output for debugging purposes. The debug logging level should never be used in a production environment.
<b>Row Level</b>	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 performed with the PDI Client or any command line tool. [Enable Logging](#) in Pentaho documentation has information about using PDI for logging.

## Logging Levels for Production, QA, and Debugging

- **Recommendation:** Use the **lowest** logging level 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) allows you to 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 works for you. Pentaho processes and stores logging within log files into a filesystem. These files can be separated into 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 Pan and Kitchen command line tools are typically used to execute PDI content, such as transformations and jobs, from outside the PDI Client. 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 guarantee quality performance.

### *Use a Single File for Root Jobs and 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. It then facilitates research when evaluating an error in a job if that server runs more than one job at a time.

### *Use a 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. Jobs show information related to entry execution, such as when an entry is started and finished. Transformations show information about start and finish time, 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. This allows you to easily diagnose your process and follow its execution. Indicate connection name and input data description for **Table input** steps. Describe the condition used as part of the name, for the **Filter Rows** step. It is important to specify a description of the stream expected, when dealing with **Dummy** steps.

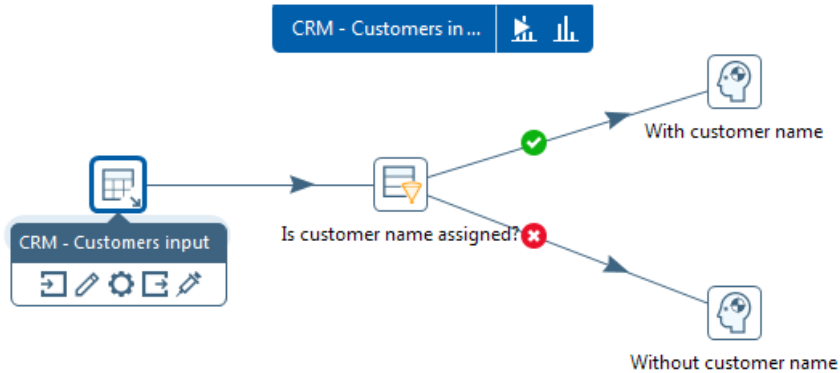


Figure 1: Naming Steps Example

- Modify the logging level to get more information out of the execution.

*Debugging Tips for Transformations and Jobs*

Developers must be able to debug processes and find the information needed regarding execution behavior. The following recommendations provide steps for three common issues.

*Identifying Errors*

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 the 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 as the first record arrives. This could be a problem within part of the data, rather than the step configuration itself.
6. Analyze the input stream of the step. Preview the transformation with or without conditions to observe data flow.

### *Investigating Unexpected Behavior*

1. Preview the transformation to find which step is causing the problem.
2. Disable the 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 steps are causing bottlenecks; this can be indicated by an output speed that is slower than the input speed. Try to identify if the issue is due to external resources, ETL design, or step slowness. These last issues are easy to identify in the PDI Client, as slow steps are represented with a dotted square around the step during the execution.
2. Replace input and output sources to discard third-party issues (e.g. network latency, database issues, etc.). Use steps like **Generate Rows** or serialize/de-serialize steps, instead.
3. Execute part of the transformation and study read rows vs write rows statistics shown 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 third-party tools to monitor DI events. The SNMP allows you to monitor the DI server or PDI process execution.



*Keep in mind that this output option is only available since Pentaho 6.0, and is recommended only if a third-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

Here are some links to information that you may find helpful while using this best practices document:

- [Best Practices - PDI Development](#)
- [Components Reference](#)
- [Enable Logging](#)
- [Kitchen Options and Syntax](#)
- [Log Rotation](#)
- [Monitoring with PDI and SNMP Traps](#)
- [Pan Options and Syntax](#)
- [PDI Logging](#)

## Finalization Checklist

This checklist is designed to be added to any implemented project that uses this collection of best practices, to verify that all items have been considered and reviews have been performed.

Name of the Project: \_\_\_\_\_

Date of the Review: \_\_\_\_\_

Name of the Reviewer: \_\_\_\_\_

Item	Response	Comments
Were you familiar with PDI logging and monitoring before beginning?	YES_____ NO_____	
Did you set up and implement log rotation appropriate for your environment?	YES_____ NO_____	
Did you follow enterprise best practices to determine the best way for you to store execution logs?	YES_____ NO_____	
Did you make sure your logging levels were appropriate for specific environments?	YES_____ NO_____	