# Pentaho Data Integration Coding Standards

* The standards presented in the document are to be used in case there is no standard coding standards in a project
* The standards can be discussed and modified accordingly as long as they are documented as part of a project
* If a customer has its own customers standards they should be adapted to the customer methodology and documented accordingly

Contents

[1 Pentaho Data Integration Coding Standards 1](#_Toc498584992)

[1.1 Naming Conventions: Transformations 2](#_Toc498584993)

[1.2 Naming Conventions: Steps 2](#_Toc498584994)

[1.3 Naming Conventions: Job 2](#_Toc498584995)

[1.4 Naming Conventions: Job Entries 2](#_Toc498584996)

[1.5 Naming Conventions: Map/Reduce Job Entry 2](#_Toc498584997)

[1.6 Naming Conventions: Variables/Parameters/Constants 2](#_Toc498584998)

[1.7 Fields 3](#_Toc498584999)

[1.8 Transformation/Job Notes 3](#_Toc498585000)

[1.9 Naming Conventions: Shell Scripts 3](#_Toc498585001)

[1.10 Naming Conventions: Log Files 3](#_Toc498585002)

[1.11 Development Complexity 3](#_Toc498585003)

## Naming Conventions: Transformations

* The transformation name should be prefixed with tr\_.
* No spaces or any special characters allowed in the name.
* Use lower case.
* Should be descriptive enough to understand what data transformation that is implemented within, e.g. tr\_parse\_customer\_data\_into\_hive.ktr

## Naming Conventions: Steps

* Define a **custom name for all steps** which signals the main function of this step.
* No steps left with the default names
* Step naming should be done using **lowercase names**
* Steps that do multiplexing (e.g.) filter, should expose the condition in the name
* In case the condition is too big, it should state the **logical name of the condition** being applied. Ex: valid customers
* Steps that do input and outputs should state the type/location of the data being processed

## Naming Conventions: Job

* A job’s name should be prefixed with jb\_.
* No spaces or any special characters allowed in the name.
* Use lower case.
* Should be **descriptive enough** to reflect the high level data flow, e.g.jb\_load\_customer\_data.kjb

## Naming Conventions: Job Entries

* Define a **custom name** for all job entries which signals the main function of this job entry.
* No job entries left with the default names.
* **Exceptions** are the START, Success and DUMMY steps
* Use **lowercase** names
* Transformation *Job Entries* and Job *Job Entries* steps should have the underlying name of the transformation/job as their name.
* In cases where the job/transformation is **referenced multiple times** a
* context should be added: Ex: tr\_common\_compute\_hive\_ddl - CUSTOMERS
* **Abort** steps should always contain the appropriate name describing the abort condition

## Naming Conventions: Map/Reduce Job Entry

* Map/Reduce jobs in the cluster can get really busy and **harder to filter** and identify using the Resource Manager
* It is recommended to use a **fixed naming convention** to **easily identify the jobs**. Example: <Project Name>: <MR Job Description>
* Consider using variables to make at least the **project name consistent**

## Naming Conventions: Variables/Parameters/Constants

* Variables and parameters are accessory to the data flows therefore it should be easy to identify which is which
* Should be **UPPERCASE** and **separated using underscores**
* The recommendation is to **prefix** variable/parameter/constant names with
  + **Variables**: VAR\_<NAME>
  + **Parameters**: PARAM\_<NAME>
  + **Properties**: PROP\_<NAME>
* **Constants** should be treated as part of the flow and should maintain a consistent format with the data fields.
* Constants in PDI are not any different from another field in the flow and should be treated as such
* All ETL projects have properties that won’t change too often, or will be environment specific
  + These properties should be stored in a common location that can be updated and potentially versioned: kettle.properties or project.properties
  + The recommendation is that the fixed property names, due to the fact that most are hierarchical, adhere to the java properties naming conversion eg.:
    - hadoop.namenode.host=localhost
    - Hadoop.namenode.port=8020

## Fields

* Fields that are present in the transformation flows should maintain as possible the metadata from the original systems that the data is being read
* New calculated fields generated from the original sources should maintain a consistent format with the original data

## Transformation/Job Notes

* Notes are a very good way to document either:
  + The overall logic that a transformation/job implements
  + Specific fixes or workarounds for issues either with the product or the business logic
  + Reasoning behind some implementation decisions might they be controversial
* There should be some common sense by the developers to not over-document the ETL
* Recommendation is to set the minimum set of notes that must be provided

## Naming Conventions: Shell Scripts

* Shell scripts to execute **Pentaho DI jobs** have to have the same name as the
* DI job plus a run\_ prefix. Example:
* **job**: jb\_dms\_master
* **shell script**: run\_jb\_dms\_master
* The **main shell script for your project** has to be named:
* run\_jb\_<project>\_master

## Naming Conventions: Log Files

Log files for a Pentaho DI job have to have the **same name as the DI job**

plus an optional suffix, e.g. the date.

Example: jb\_dms\_master.err.log

## Development Complexity

* For maintainability reasons the complexity of the developed transformations must be done within reason
* Due to the multi-threaded nature of PDI it is recommended to keep the number of steps present in a single transformation down to a reasonable level
* **The recommended maximum number of steps per transformation is currently 30**
  + There will be exceptions, these exceptions should be documented and justified
* Jobs should be broken down by logical blocks (vs. doing the full orchestration in the same job)