HITACHI
Inspire the Next

Pentaho Data Integration Naming Standards

HITACHI
Inspire the Next

Change log:

Date	Version	Author	Changes
10/11/2017	1.0	Matthew Casper	
1/5/2018	1.1	Sandra Wagner	edits
6/7/2018	1.2	Matthew Casper Megan Brown	Edits
2/6/2019	1.3	Matthew Casper Megan Brown	Update to 8.2
7/10/2019	1.4	Matthew Casper Megan Brown	Update to 8.3
2/28/2020	1.5	Matthew Casper Megan Brown	Update to 9.0

Contents

Overview

This document covers some best practices on Pentaho Data Integration (PDI). In it, you will learn PDI step naming standards and tips about how to choose appropriate steps in certain situations.

Our intended audience is PDI users or anyone with a background in ETL development who is interested in learning PDI development patterns.

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

The Components Reference in Pentaho Documentation has a complete list of supported software and hardware.

Before You Begin

Before beginning, use the following information to prepare for the procedures described in the main section of the document.

Other Prerequisites

This document assumes that you have knowledge about PDI and that you have already installed Pentaho software.

Use Cases

Use cases employed in this document include the following:

A company new to PDI needs to set forth naming standards across the team of developers to deliver consistent lines of code which could be managed by any team member and not just the person who did the development.

PDI Organization and Naming Conventions

Consistent and uniform organization, coding, and naming are important in the implementation of PDI, so that everyone using the system can find what they are looking for and understand where the information comes from and what it indicates.

You can find details on these topics in the following sections:

Job Naming

A job serves as a wrapper for one or several transformations or jobs, and provides the ability to perform database operations such as truncate, file operations such as file transfer protocol (FTP), and notifications through email. Jobs graphically represent the order in which a table or set of tables should be loaded, and any pre-processing or post-processing that should occur.

To distinguish job names from names of other objects, and to emphasize what the job is intended to accomplish, create your own naming convention to apply to all your job names wherever possible.

For example, if a directory contains more than one type of object, you could try a prefix of `j_` on all job names, and create the rest of the name according to what the job does, like this: `j_<Purpose/Action>_<Target Identifier>`.

Try using examples such as those in Table 1 for your job names. In Table 1, the first part of the job name, the `<Purpose/Action>`, is the verb of what is being done in the job, such as load, perform, build, move, delete, and so on. This piece of the name lets you know why the job exists.

The second part of the job name, the `<Target Identifier>`, is the name of the main subject of the job, such as `dim_tables` or `fact_tables`. It can also represent the names of other objects such as a database schema (`staging` or `bi` or any other), a file (`flat_file` or something similar), or a single table name (such as `fact_sales`).

Table 1: Example Job Names

Job Description	Job Name Example
Loads all of a database's dimension tables	`load_dims`
Loads all of a database's fact tables	`load_facts`
Handles load processes for a single database table	`load_sales_fact`
Main job in creating a flat file for FTP delivery	`build_ftp_transaction_file`
Validates files by comparing checksum values	`compare_checksum`
Pulls data from one area to another	`pull_oltp_to_staging`
Handles error notification	`update_on_error` `send_error_notification`

There are other alternatives you can consider to shorten job names, such as removing vowels and using `dim_tbls` instead of `dim_tables`.

Transformation Naming

A transformation contains the business rules that the source data must follow as it populates the target destination table(s) or file(s). Transformations, together with jobs, create the executable code that can be called using Kitchen, Pan, or an external scheduling tool. Transformations graphically represent a form of data flow diagram that connects the source table(s) with the target table(s).

One or more transformations may be included within a job. Examples include data warehouse table loads, fact table loads, dimension table loads, and aggregate table calculations.

To name transformations (or mappings, which use the `map_` prefix but otherwise follow these rules):

Use whichever naming standard (uppercase, lowercase) is right for your organization, but be consistent. In our example, we will use lowercase.

- Create transformation names in all lowercase.
- Use this convention for transformation or mapping naming:

```
<Purpose/Action>_<Subject/Target Identifier>
map_<Purpose/Action>_<Subject/Target Identifier>
```

- Fill in the `<Purpose/Action>` and `<Subject/Target Identifier>` parts of the name depending on what the transformation does. For example, if the transformation loads the `dim_equipment` table, try naming the transformation `load_dim_equipment`. If the transformation truncates all the dimension tables, it makes more sense to name the transformation based on that action and subject: `truncate_dim_tables`.

Table 2: Example Transformation Names

Transformation Description	Transformation Name Example
Loads the `dim_equipment` table	`load_dim_equipment`
Loads the `fact_sales` table	`load_fact_sales`
Truncates all dimension tables	`truncate_dim_tables`
Sets filename received from previous results as a variable	`set_file_name_var`
Handles inserts to the dimension tables for the `Not Applicable` or `zero` record only	`initial_insert_na_record`
Sets variables for a specific database area	`set_staging_vars` `set_dwh_vars`
Mapping that obtains different metadata properties from a text file	`map_file_properties`

As with the job naming, one way to make transformation names shorter is to remove vowels.

Job Entry Naming

A job entry name should reflect the function of the object based on the following criteria:

- An entry based on a target object (like a transformation) should inherit the target's name (like `dim_date_hour`).
- Name an entry based on an action to reflect the purpose of the action (like `delete_closure_records`).
- Always name entries in lowercase letters (or according to the consistent upper/lowercase usage of your organization).

Use the standard prefix for entries, and name them with descriptions of the operations or intent of the entry. The full list of job entry naming conventions is available in Appendix 1.

Table 3: Example Job Entry Names

Job Entry	Naming Convention	Description/Example
Transformation	Transformation name	This entry should reflect the transformation name, `load_dim_equipment`.
Simple evaluation	`se-` + purpose/attribute	If the simple evaluation is being performed on a variable determining whether the run is the initial load, the job entry name should be `se-initial_load`.
Structured Query Language (SQL)	`sql-` + action/ purpose/filename	If the SQL script contains code to drop indexes on a table, then the name can simply be `sql-drop_indexes`. You can include the table name as well, but consider the overall job entry name length.
Write to log	`wtl-` + purpose/job name	If the `load_dims` job is making note in the log of what parameter/variable values are being used at execution time, the name should be `wtl-load_dim_vars`.
Job	Job name	This should be the same name of the job being called by the job entry step, `load_dims`.
Set variables	`svar-` + purpose	If you are setting variables coming from a properties file named `staging.properties`, then the job entry name should be `svar-staging_properties`.

Transformation Step Naming

There are more than 200 different transformation steps available in developing a transformation, and the basic naming convention is a two- to seven-letter abbreviation of the step name followed by a colon and a short description identifying the step's purpose.

A step name should reflect the function of the object based on the following criteria:

- A step based on a source object (like a table or file) should inherit the source's name.
- A step based on a target object should inherit the target's name.
- Always name steps in lowercase letters (or according to the consistent upper/lowercase usage of your organization).

Use the standard prefix for steps, and name them with descriptions of the operations or intent of the entry. The full list of step naming conventions is available in Appendix 2.

Table 4: Example Step Names

Step Type	Naming Convention	Description/Example
Add constants	`ac-`+ attribute/subject name	Name this step as `ac-`+ the name of the attribute being generated, or, if there are many attributes, a subject area name would be sufficient.
Add sequence	`aseq-`+ attribute name for which the sequence is being generated	If the sequence is being generated for `column_product_key`, then the step should be named `aseq-product_key`.
Database lookup	`dblkp-`+ attribute name for which lookup is being performed	If the lookup is being performed on `product_key` using the table `DIM_PRODUCT`, then name the step `dblkp-product_key`.
Filter rows	`fr-`+ attribute name for which the filter is being applied	If the filter step is based on a filter condition for the state attribute, name the step `fr-state`.
Get system info	`gsi-`+ attribute name for which information is being selected	Name this step as `gsi-`+ the name of the attribute being generated. If the `system_date` is being pulled, name the step `gsi-system_date`.
Insert/Update	`iu-`+ target/entity name for which the update strategy is being defined	If the insert/update strategy is being defined for the table `customer`, name the step `iu-customer`.
Select values	`sv-`+ purpose + attribute	Use this step to clean up a data stream so that only the needed columns are included or so the columns have the correct name or data type. If you are renaming attribute `reason_1` to `reason`, name the step `sv-rename_reason`.
Table input	`ti-`+ source/subject name	If the source data is defined on table `dim_product`, then name the step `ti-dim_product`. If many tables are used, provide a subject name that represents the data being pulled, like `ti-transaction_data`.
Table output	`to-`+ the name of the target	These step names should represent the target table. If you are inserting records into the `customer` table, name the step `to-customer`.

Related Information

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

- Components Reference

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. (Compose specific questions about the topics in the document and put them in the table.)

Name of the Project:____

Date of the Review:____

Name of the Reviewer:____

Item	Response	Comments
Did you set up a clearly named folder structure?	YES_______ NO_______	
Did you systematically name folders, jobs, transformations, and steps?	YES_______ NO_______	

Appendix 1: Job Entry Naming Conventions

Here is the complete list of job entry naming conventions:

Table 5: Job Entry Naming Conventions

Job Entry Name	Naming Convention
Abort job	`'abrt-' + purpose/reason`
Add filenames to result	`'afntr-' + file name/purpose`
Amazon EMR Job Executor	`'aemrje-' + job name`
Amazon Hive Job Executor	`'ahje-' + job name`
Build model	`'bmdl-' + job name`
Bulk load from MySQL into file	`'blmysqlf-' + file name/purpose`
Bulk load into Amazon Redshift	`'blkar' + target`
Bulk load into MSSQL	`'blmssql-' + target`
Bulk load into MySQL	`'blmysql-' + target`
Bulk load into Snowflake	`'blksf' + target`
Check DB connections	`'chkdbc-' + connection name/purpose`
Check files locked	`'chkfl-' + file/subject area/purpose'`
Check if a folder is empty	`'chkfldr-' + folder name/purpose`
Check if connected to repository	`'chkrepo-' + repo name/purpose`
Check if XML file is well formed	`'chkxml-' + xml name/purpose`
Check webservice availability	`'chkws-' + webservice name/purpose`
Checks if files exist	`'chkfe-' + file name/subject area`
Columns exist in a table	`'colex-' + column name/table name/subject area`
Compare folders	`'cfldrs-' + purpose/subject area`
Convert file between Windows and Unix	`'cnvrtf-' + file name/purpose`
Copy files	`'cf-' + file name/subject area/purpose`
Create a folder	`'cfldr-' + folder name`
Create file	`'cf-' + file name`
Create Snowflake warehouse	`'csfw' + target`
Decrypt files with PGP	`'dpgp-' + file name/subject area/purpose`
Delete file	`'df-' + file name`
Delete filenames from result	`'dffr-' + file name/subject area/purpose`
Delete files	`'dfls-' + file name/subject area/purpose`
Delete folders	`'dfldrs-' folder name/subject area/purpose`

Job Entry Name	Naming Convention
Delete Snowflake warehouse	`'dsfw' + target`
Display msgbox info	`'msgbox-' + purpose`
DTD Validator	`'dtdval-' + dtd file name/xml file name`
Dummy	`'dmmy-' + purpose`
Encrypt files with PGP	`'epgp-' + file name/subject area/purpose`
Evaluate files metrics	`'evalfm-' + file name/subject area/purpose`
Evaluate rows number in a table	`'evalrnt-' + table name/purpose`
Example job (deprecated)	`'exjb-' + job name`
Export repository to XML file	`'erepo-' + xml file name`
File compare	`'fc-' + files/subject area/purpose`
File exists	`'fe-' + file name/subject area/purpose`
FTP delete	`'ftpd-' + ftp server name/file name/subject area/purpose`
Get a file with FTP	`'gftp-' + file name`
Get a file with FTPS	`'gftps-' + file name`
Get a file with SFTP	`'gsftp-' + file name`
Get mails (POP3/IMAP)	`'gmail-' + purpose/subject area`
Google BigQuery loader	`'gbql-' + target`
Hadoop copy files	`'hadcp-' + file name/purpose/subject area`
Hadoop job executor	`'hadje-' + job name`
HL7 MLLP acknowledge	`'hl7ack-' + server name/message/purpose`
HL7 MLLP input	`'hl7i-' + source`
HTTP	`'http-' + purpose/webpage name`
JavaScript	`'js-' + purpose`
Job	Job Name
Mail	`'mail-' + purpose`
Mail validator	`'mailv-' + purpose`
Modify Snowflake warehouse	`'msfw' + target`
Move files	`'mf-' + file name/purpose/subject area`
MS Access bulk load (deprecated)	`'msabl-' + target`
Oozie job executor	`'oozje-' + job name`
Palo cube create (deprecated)	`'pcc-' + cube name`
Palo cube delete (deprecated)	`'pcd-' + cube name`
Pentaho MapReduce	`'pmr-' + purpose`
Pig script executor	`'pse-' + script name/purpose`

Job Entry Name	Naming Convention
Ping a host	`'ping-' + host name/purpose`
Process result filenames	`'prfn-' + file name/purpose/subject area`
Publish model	`'pubm' + model name`
Put a file with FTP	`'pftp-' + file name`
Put a file with SFTP	`'psftp-' + file name`
Send information using syslog	`'syslog-' + server name/message purpose`
Send Nagios passive check	`'nagios-' + host name/message purpose`
Send SNMP trap	`'snmp-' + server name/message purpose`
Set variables	`'svar-' + purpose`
Shell	`'shll-' + script name/purpose`
Simple evaluation	`'se-' + purpose/attribute`
Spark submit	`'sprks-' + job name`
SQL	`'sql-' + action/purpose/file name`
Sqoop export	`'sqpex-' + target`
Sqoop import	`'sqpim-' + source`
Start	`Start`
Start a PDI cluster on YARN	`'strtyarn-' + cluster schema/purpose`
Start Snowflake warehouse	`'strtsfw' + target`
Stop a PDI cluster on YARN	`'stpyarn-' + cluster schema/purpose`
Stop Snowflake warehouse	`'stpsfw' + target`
Success	`Success`
Table exists	`'tble-' + table name`
Talend Job Execution (deprecated)	`'talje-' + job name`
Telnet a host	`'telh' + hostname`
Transformation	`Transformation name`
Truncate tables	`'trunc-' + purpose/schema name`
Unzip file	`'unzip-' + file name/purpose`
Upload files to FTPS	`'uftps-' + file name/purpose/subject area`
Verify file signature with PGP	`'vpgp-' + file name/purpose`
Wait for	`'wait-' + purpose`
Wait for file	`'waitf-' + file name`
Wait for SQL	`'waits-' + purpose`
Write to file	`'wtf-' + file name`

Job Entry Name	Naming Convention
Write to log	`'wtl-' + purpose/job name`
XSD validator	`'xsdval-' + xml file name`
XSL transformation	`'xlst-' + xml file name/output file name'`
Zip file	`'zip-' + file name`

Appendix 2: Step Naming Conventions

Here is the complete list of step naming conventions:

Table 6: Step Naming Conventions

Step Name	Naming Convention
Abort	`'abrt-' + purpose/reason`
Add a checksum	`'chksum-' + subject name/purpose`
Add constants	`'ac-'+ attribute/subject name`
Add sequence	`'aseq-'+ attribute name for which sequence is being generated`
Add value fields changing sequence	`'chgseq-' + attribute/purpose`
Add XML	`'axml-' + subject name/reason`
AMQP consumer	`'amqpc-' + source`
AMQP producer	`'amqpp-' + target`
Analytic query	`'aq-' + subject name/purpose`
Annotate stream	`'anns-' + purpose/reason`
Append streams	`'as-' + subject name/purpose`
Arff output	`'arffo-' + target`
Automatic documentation output	`'ado-' + target`
Avro input	`'avroi-' + source/subject name`
Avro output	`'avroo-' + target/subject name`
Block this step until steps finish	`'bsusf-' + purpose/subject name`
Blocking step	`'bs-' + purpose/subject name`
Calculator	`'calc-' + purpose`
Call DB procedure	`'dbproc-' + procedure name`
Call endpoint	`'endpt-' + purpose/subject name`
Cassandra input	`'cassi-' + source`
Cassandra output	`'casso-' + target`
Change file encoding	`'chgfe-' + purpose/subject name`
Check if file is locked	`'chkfl-' + file name`
Check if webservice is available	`'chkwsa-' + webservice name`
Clone row	`'cr-' + purpose`
Closure generator	`'cg-' + Hierarchy`
Column exists	`'ce-' + column name`

Step Name	Naming Convention
Combination lookup/update	`'clkpu-' + table name`
Concat fields	`'cf-' + name of target/purpose`
Copy rows to result	`'crtr-' + purpose/subject name`
Copybook Input	`'cbi' + source name`
CouchDB input	`'cdbi-' + source name`
Credit card validator	`'ccval-' + field name/card type`
CSV file input	`'csvi-' + source name`
Data grid	`'dg-' + source/subject name`
Data validator	`'dval-' + purpose/subject name`
Database join	`'dbjn-'+purpose`
Database lookup	`'dblkp-'+ attribute name for which lookup is being performed`
Delay row	`'delr-' + purpose/subject name`
Delete	`'del-' + target name`
De-serialize from file	`'dsff-' + name of object/purpose`
Detect empty stream	`'des-' + subject name/purpose`
Dimension lookup/update	`'dimlu-' + dimension to be updated`
Dummy (do nothing)	`'dmmy-' + purpose/subject name`
Dynamic SQL row	`'dsqlr-' + purpose/table name`
EDI to XML	`'edixml-' + purpose/target`
Elasticsearch bulk insert	`'elsbi-' + target name`
Email messages input	`'emsgi-' + source/subject name`
ESRI shapefile reader	`'esri-' + source/subject name`
ETL metadata injection	`'emi-' + purpose`
Example step (deprecated)	`'exst-' + subject name/purpose`
Execute a process	`'execp-' + process name`
Execute row SQL script	`'exrsql-' + purpose/subject name`
Execute SQL script	`'exsqls-' + script name`
File exists	`'fe-' + filename`
Filter rows	`'fr-'+ attribute name for which filter is being applied`
Fixed file input	`'fxdfi-' + file name`
Formula	`'frmla-' + attribute/subject name`
Fuzzy match	`'fzzym-' + purpose/subject name`
Generate random credit card	`'grcc-' + purpose/subject name`

Step Name	Naming Convention
Generate random value	`'grval-' + attribute/subject name`
Generate rows	`'gr-' + attribute/subject name`
Get data from XML	`'gdxml-' + source`
Get file names	`'gfn-' + file/purpose`
Get files from result	`'gffr-' + files/purpose`
Get files rows count	`'gfrc-' + file/purpose`
Get ID from slave server	`'gidfss-' + purpose/slave server name`
Get records from stream	`'grfs-' + source/subject`
Get repository names	`'grnms-' + repo name/purpose`
Get rows from result	`'grfr-' + purpose/subject area`
Get session variables	`'gsvars-' + purpose/variable name`
Get subfolder names	`'gsfnms-' + purpose/directory name`
Get system info	`'gsi-'+ attribute name for which information is being selected`
Get table names	`'gtbls-' + schema name/purpose`
Get variables	`'gv-' + purpose/target table name`
Google Analytics	`'gglea-' + purpose`
Greenplum bulk loader (deprecated)	`'gpbl-' + target`
Greenplum load	`'gpl-' + target`
Group by	`'grp-' + attribute/subject name`
GZIP CSV input	`'gzcsvi-' + source`
Hadoop file input	`'hdpfi-' + source`
Hadoop file output	`'hdpfo-' + target`
HBase input	`'hbsi-' + source`
HBase output	`'hbso-' + target`
HBase row decoder	`'hbsrd-' + purpose/subject area/table name`
HL7 input	`'hl7i-' + source`
HTTP client	`'httpc-' + purpose`
HTTP post	`'httpp-' + purpose/web page`
IBM Websphere MQ consumer (deprecated)	`'ibmc-' + source`
IBM Websphere MQ producer (deprecated)	`'ibmp-' + target`
Identify last row in a stream	`'idlr-' + purpose/field name`

Step Name	Naming Convention
If field value is null	`'ifvaln-' + subject name`
Infobright loader	`'ibldr-' + target`
Ingres VectorWise bulklLoader	`'vwbldr-' + target`
Injector	`'inj-' + purpose`
Insert/update	`'iu-' + target/entity name for which update strategy is being defined`
Java filter	`'javaf-' + purpose/field name`
JMS consumer	`'jmsc-' + source`
JMS producer	`'jmsp-' + target`
Job executor	`'je-' + target job name`
Join rows (cartesian product)	`'jr-' + purpose`
JSON input	`'jsoni-' + source`
JSON output	`'jsono-' + target`
Kafka consumer	`'kafc-' + source`
Kafka producer	`'kafp-' + target`
Kinesis Consumer	`'kinc' + source`
Kinesis Producer	`'kinp' + target`
Knowledge Flow	`'kfl-' + source/purpose`
LDAP input	`'ldapi-' + source`
LDAP output	`'ldapo-' + target`
LDIF input	`'ldifi-' + source`
Load file content in memory	`'lfcm-' + source`
LucidDB streaming loader (deprecated)	`'ldbsl-' + target`
Mail	`'mail-' + purpose/subject name`
Mail validator	`'mailv-' + purpose`
Mapping (sub-transformation)	`'m-' + purpose`
Mapping input specification	`'mis-' + purpose/source`
Mapping output specification	`'mos-' + purpose/target`
MapReduce input	`'mapri-' + source`
MapReduce output	`'mapro-' + target`
Memory group by	`'mgb-' + purpose/field name`
Merge join	`'mj-'+ purpose`
Merge rows (diff)	`'mr-' + purpose`
Metadata structure of stream	`'mdss-' + purpose`

Step Name	Naming Convention
Microsoft Access input	`'msai-' + source`
Microsoft Access output	`'msao-' + target`
Microsoft Excel input	`'msxlsi-' + source`
Microsoft Excel output	`'msxlso-' + target`
Microsoft Excel writer	`'msxlsw-' + target`
Modified JavaScript value	`'mjsv-'+ attribute name being generated`
Mondrian input	`'mndrni-' + source`
MonetDB Agile Mart	`'mdbam-' + target`
MonetDB bulk loader	`'mdbbl-' + target`
MongoDB input	`'mdbi-' + source`
MongoDB output	`'mdbo-' + target`
MQTT consumer	`'mqttc-' + source`
MQTT producer	`'mqttp-' + target`
Multiway merge Join	`'mwmj-' + purpose`
MySQL bulk loader	`'mysqlbl-' + target`
Null if	`'null-' + field name/purpose`
Number range	`'nbrr-' + field name/purpose`
OLAP input	`'olapi-' + source`
Oracle bulk loader	`'orabl-' + target`
ORC input	`'orci-' + source`
ORC output	`'orco-' + target`
Output steps metrics	`'osm-' + subject name`
Palo cell input (deprecated)	`'paloi-' + source`
Palo cell output (deprecated)	`'paloo-' + target`
Palo dim input (deprecated)	`'palodi-' + source`
Palo dim output (deprecated)	`'palodo-' + target`
Parquet input	`'parqi' + source`
Parquet output	`'parqo' + target`
Pentaho reporting output	`'pro-' + target`
PGP decrypt stream	`'pgpd' + source`
PGP encrypt stream	`'pgpe' + target`
PostgreSQL bulk loader	`'psqlbl-' + target`
Prioritize streams	`'ps-' + name of key field/purpose`
Process files	`'pfiles-' + source/directory/subject name`

Step Name	Naming Convention
Properties output	`'po-' + name of target file/purpose`
Property input	`'pi-' + source`
Python Executor	`'pe-' + script name/purpose`
Query HCP	`'qhcp' + source`
R script executor	`'rscrpt-' + script name`
Read metadata from Copybook	`'rmcp' + source`
Read metadata from HCP	`'rmhcp' + source`
Regex evaluation	`'re-' + purpose`
Replace in string	`'ris-' + file/purpose`
Reservoir sampling	`'rs-' + purpose`
REST client	`'rest-' + source/purpose`
Row denormaliser	`'rdnml-' + source/subject name`
Row flattener	`'rf-' + purpose/source field`
Row normaliser	`'rnml-' + source/subject name`
RSS input	`'rssi-' + source`
RSS output	`'rsso-' + target`
Rules accumulator	`'racc-' + source/purpose`
Rules executor	`'rexe-' + source/purpose`
Run SSH commands	`'rssh-' + source/purpose`
S3 CSV input	`'s3csvi-' + source`
S3 file output	`'s3fo-' + target`
Salesforce delete	`'slsfd-' + target`
Salesforce input	`'slsfi-' + source`
Salesforce insert	`'slsfinsrt-' + target`
Salesforce update	`'slsfu-' + target`
Salesforce upsert	`'slsfupsrt-' + target`
Sample rows	`'smplr-' + purpose`
SAP HANA bulk loader	`'sapbl-' + target`
SAP input (deprecated)	`'sapi-' + source`
SAS input	`'sasi-' + source`
Script (deprecated)	`'scrpt-' + purpose`
Secret key generator	`'skgen-' + purpose/key field`
Select values	`'sv-'+ purpose`
Send message to syslog	`'smsgsl-' + target/purpose`

Step Name	Naming Convention
Serialize to file	`'stf-' + name of target/purpose`
Set field value	`'sfv-' + field name/purpose`
Set field value to a constant	`'sfvc-' + field name/purpose`
Set files in result	`'sfnr-' + filename field/purpose`
Set session variables	`'ssvar-' + purpose/variable name`
Set variables	`'svar-' + purpose`
SFTP put	`'sftpp-' + target/purpose`
Shared dimension	`'sdim' + source`
Simple mapping (sub-transformation)	`'sm-' + target/purpose`
Single threader	`'sthr-' + target/purpose`
Socket reader	`'scktr-' + source`
Socket writer	`'scktw-' + target'`
Sort rows	`'sr-' + purpose`
Sorted merge	`'sm-' + purpose`
Split field to rows	`'sftr-' + purpose`
Split fields	`'sf-' + purpose`
Splunk input	`'splnki-' + source`
Splunk output	`'splnko-' + target`
SQL file output	`'sqlfo-' + target`
SSTable output	`'sstblo-' + target`
Stream lookup	`'slkp-' + attribute name/purpose`
String operations	`'so-' + name of key field/purpose`
Strings cut	`'sc-' + purpose`
Switch/case	`'swcs-' + process being evaluated`
Symmetric cryptography	`'symc-' + result field name/purpose`
Synchronize after merge	`'syncham-' + target`
Table Agile Mart	`'tblam-' + purpose`
Table compare	`'tblc-' + target/purpose`
Table exists	`'tble-' + target`
Table input	`'ti-' + source/subject name`
Table output	`'to-' + name of the target`
Teradata Fastload bulk loader	`'tdbl-' + target`
Teradata TPT bulk loader	`'tdiubl-' + target`

Step Name	Naming Convention
Text file input	`'tfi-' + file name/purpose`
Text file output	`'tfo-' + file name`
Transformation executor	`'te-' + transformation name/purpose`
Unique rows	`'ur-' + field name/purpose`
Unique rows (HashSet)	`'urh-' + field name/purpose`
Univariate statistics	`'unvstats-' + field name/purpose`
Update	`'updt-' + name of target`
User defined Java class	`'udjc-' + purpose/field name`
User defined Java expression	`'udje-' + purpose`
Value mapper	`'vm-' + subject name/purpose`
Vertica bulk loader	`'vbl-' + target`
Web services lookup	`'wslkp-' + target/purpose`
Weka Forecasting	`'wekaf-' + subject name/purpose`
Weka Scoring	`'wekas-' + subject name/purpose`
Weka Time Series Forecasting	`'wtsf' + source/purpose`
Write metadata to HCP	`'wmhcp' + target/purpose`
Write to log	`'wlog-' + purpose`
Xbase input	`'xbsi-' + source`
XML input stream (StAX)	`'xmlis-' + source`
XML join	`'xmlj-' + target/purpose`
XML output	`'xmlo-' + name of the target`
XSD validator	`'xsdv-' + purpose/source file`
XSL transformation	`'xslt-' + source/purpose`
YAML Input	`'yamli-' + source`
Zip file	`'zip-' + target`