Best Practices - PDI Development

This page intentionally left blank.

Contents

This page intentionally left blank.

Overview

This document is designed to give developers and administrators best practices around the set up and configuration of directories to be used for Pentaho Data Integration (PDI) development and execution.

Keep these Pentaho Architecture principles in mind while you are working through this document:

1. Architecture is important, above all else.
2. Platforms are always evolving: sometimes you will have to think creatively.

Some of the things discussed here include folder structures for workstations and servers, configuration, and migrating content.

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

Software	Version
Pentaho Data Integration	4.x, 5.x, 6.x, 7.x

Directory / Folder Structures

The development and deployment of PDI-based solutions follows a convention that places all project-related development items into one folder. That folder and its content can be migrated from environment to environment, as appropriate.

Client/Workstation Folder Structure

Set up one main folder for each project using PDI. Each project and variant should have its own unique `KETTLE_HOME` variable that is not shared and set for each execution as described in next section. Each project should have the following folders:

- **Content**: Transformations and jobs will be stored in this folder. Root folders of `home` and `public` should always be used.
- **Config**: Project properties and other configuration files
- **Input**: Files that are input to the solution
- **Output**: Files that are produced as part of this solution
- **Env**: Environment folder where each unique environment variation is stored. The following variants will help you create different folders within one project:
  - **Pentaho Version**: Each Version should have its own folder.
  - **Content Storage**: If you are connecting to files or a repository.
  - **Server Environment**: Dev, Testing, and Production should all have their own `env` variant.
  - **Container**: Where the variant is intended to execute (workstation, carte, or DI server).

```
project1
-content
--public
---project1
--home
---admin
---user1

-config
--project1.properties

-input
-output

-env
--ws_files_v7
---.kettle
----kettle.properties
----repositories.xml
----shared.xml
---spoon.bat/sh

--ws_files_v6
---.kettle
----kettle.properties
----repositories.xml
----shared.xml
---spoon.bat/sh
```

Server Folder Structure

The Server running Pentaho and supporting multiple projects at once within one JVM should have the following directories:

Note that logs are collected centrally for ease of monitoring, archive, and maintenance. Parameters, inputs, and outputs are all collected per project.

Server Install (/opt/pentaho)

```
server
-data-integration-server or -pentaho-server (if using Pentaho 7.0)
--pentaho-solutions
---system
----slave-config.xml
-data-integration

User Home Directory (use script in the env folder)
Nothing is actually here (bad practice)

Project Directory (/projects/project1)
content
config
input
output
env

Central Log Directory (/projects/log)
-project1
--job1datetime.log
--job2datetime.log
-project2
--job3datetime.log
--job3datetime.log
```

Development for Project

When developing transformations for `project1`, run the `bat/sh` script that is unique to the type of connection that you have (local, dev, prod, etc.). Each connection should launch spoon in its own kettle home. Setting `KETTLE_HOME` this way allows all connection specific information to be self-contained in this `KETTLE_HOME` directory. When you switch to talking to each repository, use another separate `KETTLE_HOME` with different configurations.

Here is a sample `spoon.bat` in `./projects/project1/env/ws_files_v51`

```
set KETTLE_HOME=U:/projects/project1/env_ws_files_v51
call U:\myfiles\pdi-ee-client-5.1\data-integration\Spoon.bat
```

Configuration

The configuration for the kettle properties only needs to be migrated for the first time a server is stood up. Each time a new project is added or modifications made, then `project.properties` file must be synchronized with the server. When migrating a DEV properties file to the server, administrators must replace the DEV references with non-DEV references as appropriate.

Kettle Properties

Each JVM that runs Pentaho Data Integration should source the default kettle properties for that environment. Each `kettle.properties` should define at least these minimum global settings. Additional project specific setting should be in the `project.properties` files, not the global `kettle.properties` file.

```
KETTLE_CHANNEL_LOG_SCHEMA=
KETTLE_CHANNEL_LOG_DB=
KETTLE_CHANNEL_LOG_TABLE=
KETTLE_JOB_LOG_DB=
KETTLE_JOB_LOG_SCHEMA=
KETTLE_JOB_LOG_TABLE=
KETTLE_JOBENTRY_LOG_SCHEMA=
KETTLE_JOBENTRY_LOG_DB=
KETTLE_JOBENTRY_LOG_TABLE=
KETTLE_TRANS_LOG_SCHEMA=
KETTLE_TRANS_LOG_DB=
KETTLE_TRANS_LOG_TABLE=
KETTLE_STEP_LOG_SCHEMA=
KETTLE_STEP_LOG_DB=
KETTLE_STEP_LOG_TABLE=
KETTLE_TRANS_PERFORMANCE_LOG_DB=
KETTLE_TRANS_PERFORMANCE_LOG_SCHEMA=
KETTLE_TRANS_PERFORMANCE_LOG_TABLE=

KETTLE_REDIRECT_STDERR=Y
KETTLE_REDIRECT_STDOUT=Y

PROJECT_DIR=\projects
```

Project Properties

Each project running under Pentaho Data Integration should have project specific settings in the `project.properties` files, not the global `kettle.properties` file. Use a transformation to handle the `project.properties`.[1]

```
ActiveProject.Home=$PROJECT_DIR\$PROJECT_NAME
project1_target_hostname=192.168.1.1
project1_target_db=dbname
project1_target_user=admin
project1_target_pass=password
```

Content Migration Overview

There are generally 2 modes of operation depending on your environment and usage:

- **External repository** (SVN, git, etc...) - With this method, you would use external tools and methods to migrate the content to the server. Typically, a checkout of the project.
- **Pentaho Repository** (Enterprise, Database, File) - When migrating from a repository, the content must be exported from the workstation and imported into the server repository.

Export Content

- **Graphical Export** - The entire repository or individual folders can be exported.
- **Command Line** - A command line can be used for the export step where many options are available. This is often used in script automation scenarios.
- **Job step** - There is a job step that can be used to selectively export repository contents based on variables and parameters.

Import Content

- **Graphical Import** - The contents of a prior repository export can be imported through the Spoon Repository menu. To keep the directories correct, always choose the "/" node for import.
- **Command Line** - A command line can be used for the import step where many options are available. This is often used in script automation scenarios.

[1] Roland Bouman's blog post on Managing kettle job configurations has more details on working with project.properties files in kettle.

Related Information

Here are some links to related information that you may find useful: .

- Pentaho Documentation: Import and Export PDI Content
- Roland Bouman's Blog- Managing Kettle job configuration

Best Practice Check List

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
Did you set up the workstation folder structure?	YES ___ NO ___	
Did you set up the server folder structure?	YES ___ NO ___	
Did you configure the Kettle properties?	YES ___ NO ___	
Did you configure the project properties?	YES ___ NO ___	
Have you migrated your content?	YES ___ NO ___	