HITACHI
Inspire the Next

Pentaho Data Integration (PDI) Techniques: Dividing Large Repositories

This page intentionally left blank.

Contents

This page intentionally left blank.

Overview

You may have initially created a single Pentaho Data Integration (PDI) repository to maintain multiple environments – development, quality assurance, stage, production – when first installing PDI. Following that, you might have divided this single repository into different environments using nested folders.

Over time, this single repository may grow to a size that negatively impacts performance. You may also have found that management of a single repository is cumbersome, even if all environments are non-production. Using PDI objects to selectively export and import folders and dividing the repository can be very efficient.

This document addresses these problems by detailing an automated method to improve performance by segmenting a single repository into several smaller repositories using PDI.

Software	Version(s)
Pentaho	6.x, 7.x, 8.x, 9.0

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

Before You Begin

This document assumes that you are familiar with PDI and its repositories, Spoon (PDI client), and the command line interface.

Use Case: Divide a Large PDI Repository for Better Performance

Wade is dealing with a single PDI repository that he'd created months ago. Although the repository can be divided into different environments, it has since increased to a size that has a negative impact on performance. He has also discovered that it takes tough management to keep the repository operating properly.

We recommend exporting the entire repository, and then reimporting it into new separate target environments. PDI objects will also help him, by giving him the option to choose which folders he wants to export/import.

Use PDI to Divide the Repository

There are a few steps required to use PDI to divide your large repository. Before you begin, make sure to create a backup of your PDI repository.

This document uses examples based on sample files to help illustrate the processes. These samples, while unsupported by Pentaho, can serve as a template you can alter for use in your environment.

You can find more information on the following steps in these sections:

- Segmenting the Repository Folders
- Exporting the Repository Folders to Files
- Import the Repository Folders to the New Repository

Segmenting the Repository Folders

You will need to complete a process to match existing folders to new repositories must be completed, regardless of which option you have chosen to divide the repository. Using an automated approach, this information is listed in a file.

For example, you can use a file named `repository_folder_list.txt` that includes a single field, `folder_name`. This field includes each repository folder that you want to move to a new repository. The example text below would move three `dev` folders from a single repository to a new development repository:

```
folder_name
/home/dev/application1/
/home/dev/application2/
/home/app3/
```

Exporting the Repository Folders to Files

Once you have created the list of repository folders, you can automatically export the repository folders themselves to XML files and folders in a file system. PDI includes an **Export repository to XML file** job entry specifically designed for this task.

The following example shows a fully parameterized **Export repository to XML file** job entry. As currently configured, it will create one set of export folders and an XML file for every repository folder defined in the `repository_folder_list.txt` file.

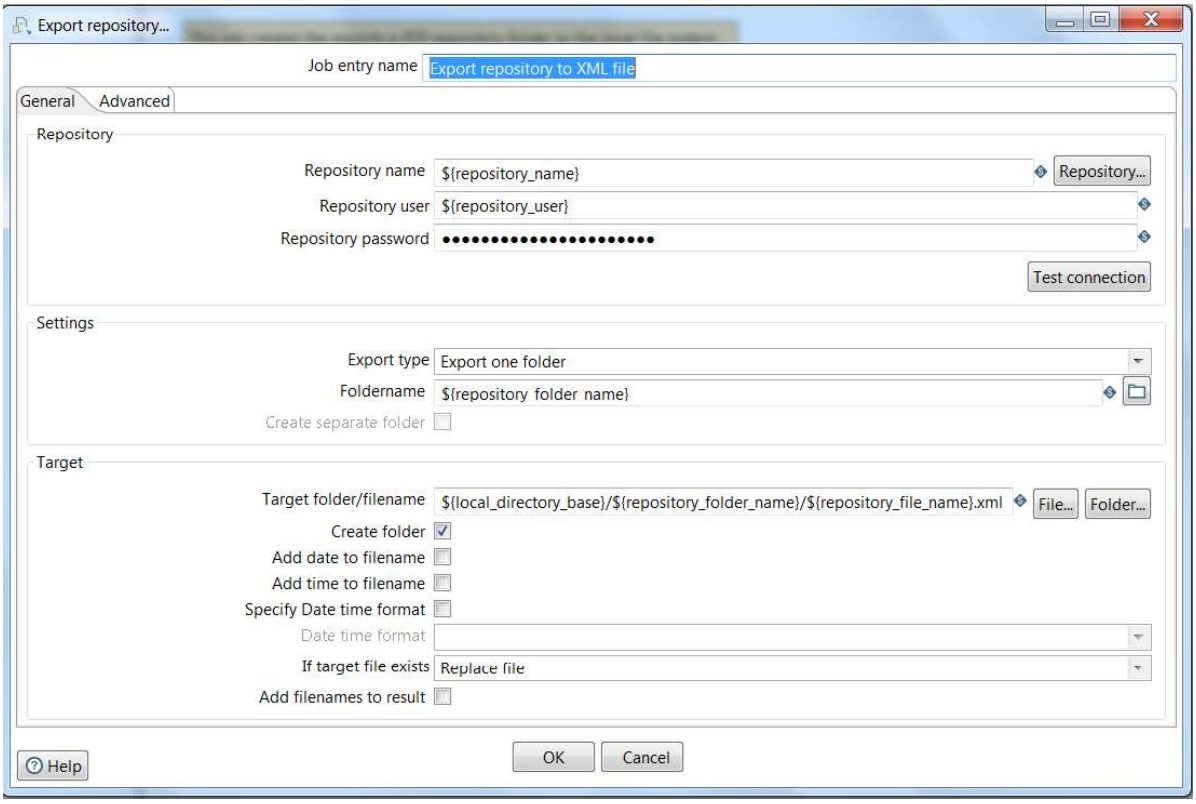

Figure 1: ***Export repository to XML file*** *Job Entry*

To fully automate the process of creating export files for all the folders listed in `repository_folder_list.txt` file, a few jobs and transformations must be built around the **Export repository to XML file** job-entry to perform the following tasks:

1. Check for a `repository_folder_list.txt` file. If it exists, then read the list of PDI repository folders to be exported.
2. For each folder listed in the file, export the repository folder to the local file system using the **Export repository to XML file** job-entry.

To make the export process flexible, the following five parameters must be provided:

Parameter	Definition
`file_name`	The file name for the file that lists the repository folders to be exported.
`local_directory_base`	The local directory where the repository folder files will be written.
`repository_name`	The name of the source repository.
`repository_user`	The name of the source repository user.
`repository_password`	The password for the source repository user.

Import the Repository Folders to the New Repository

Once the repository folders have been exported from the source repository to the XML files on the file system, they can be automatically imported into the new repository.

Note: this section assumes that new repositories have already been created in your new PDI environments.

PDI includes script files - `import.sh` for Linux, `import.bat` for Windows - specifically designed for this task.

The following example shows a fully parameterized command that will import one repository folder for each entry in the `repository_folder_list.txt` file by calling `import.sh`:

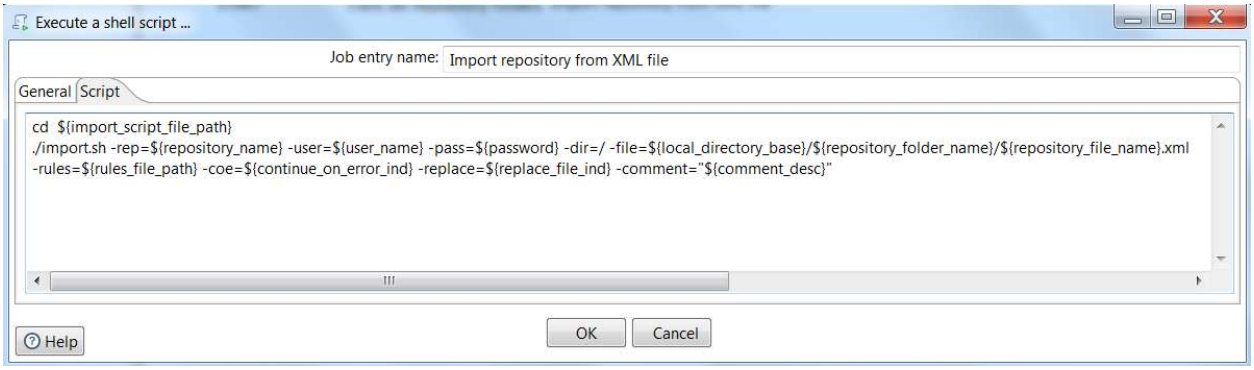

*Figure 2: Execute `Import.sh` Using the **Shell** Job Entry*

Here is the command used in the **Shell** job entry:

```
cd  ${import_script_file_path}

./import.sh -rep=${repository_name} -user=${user_name} -pass=${password} -
dir=/ -
file=${local_directory_base}/${repository_folder_name}/${repository_file_na
me}.xml -rules=${rules_file_path} -coe=${continue_on_error_ind} -
replace=${replace_file_ind} -comment="${comment_desc}"
```

A few jobs and transformations must be built around the import script to perform the following tasks, to fully automate the process of importing all the folders listed in `repository_folder_list.txt`:

1. Check for a `repository_folder_list.txt` file.
2. If it exists, then read the list of PDI repository folders to be imported.
3. For each folder listed in the file, import the repository folder from the local file into the target repository.

To make the import process flexible across environments, provide the following job parameters:

Parameter	Definition
`file_name`	The file name for the file that lists the repository folders to be imported. Use the same file used to create export list.
`import_script_file_path`	The path to the `import.sh` or `import.bat` file - does not include file name.
`local_directory_base`	The local directory where the repository folder files will be written.
`local_directory_base`	The password for the `username` you specified with user.
`repository_name`	The name of the enterprise or database repository to import into.
`rules_files_path`	The path to the rules file, including full directory and file name.
`user_name`	The repository `username` you will use for authentication.

The following optional job parameters may also be helpful:

Parameter	Definition
`comment_desc`	The comment that will be set for the new revisions of the imported transformations and jobs.
`continue_on_error_ind`	**Continue on error**, ignoring all validation errors. Defaults to `false`.
`replace_file_ind`	Set to `Y` to replace existing transformations and jobs in the repository (creates a new version if versioning is turned on). Default value is `N`.

Finally, an `import-rules.xml` file must be created and placed in the path specified in the `rules_file_path` parameter.

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
Did you use the automated approach to locate the process for matching existing folders to new repositories?	YES_____ NO_____	
Did you build the jobs and transformations around the **Export repository to XML file** job entry to perform the tasks needed for automating the process of exporting folders to files?	YES_____ NO_____	
Did you build the jobs and transformations around the import script to fully automate the process of importing folders?	YES_____ NO_____	