Table of Contents

Guided Demo 1 – Launch & Customize PDI 2
Guided Demo 2 – Creating a 'Hello World' Transformation 8
Exercise 1 – Generate Rows, Sequence, Select Values 14
Guided Demo 3 – Error Handling & Basic Logging 21
Guided Demo 4 – Saving a Transformation in the Repository 25
Guided Demo 5 – Combining Several Inputs into One Output 29
Guided Demo 6 – Creating kettle.properties Variables 38
Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case 43
Exercise 2 Advanced – CSV Input to Multiple Text Output Using Switch/Case 52
Exercise 3 – Serializing Multiple Text Files 54
Exercise 3 Advanced – Serializing Multiple Text Files 59
Exercise 4 – De-serializing a File 60
Exercise 4 Advanced – De-Serializing Multiple Text Files 67
Guided Demo 7 – Connections & the Database Explorer 68
Exercise 5 – Reading & Writing to Database Tables 74
Exercise 5 Advanced – Reading & Writing to Database Tables 84
Guided Demo 8 – Data Cleansing 87
Exercise 6 – Input with Parameters & Table Copy Wizard 98
Exercise 6 Advanced – Input with Parameters & Table Copy Wizard 109
Exercise 7 – Parallel Processing 110
Exercise 7 – Parallel Processing, Continued 111
Exercise 7 – Parallel Processing, Continued 112
Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields' 119
Exercise 8 – Lookups & Data Formatting 129
Guided Demo 10 – Creating Summary Fields Using Group By 137
Exercise 9 – Calculating & Aggregating Order Quantity 143
Exercise 9 Advanced – Calculating & Aggregating Order Quantity 153
Exercise 10 – Loading JVM Data into a Table 155
Exercise 10 Advanced – Loading JVM Data into a Table 165
Exercise 11 – Using the Pentaho Enterprise Repository 167
Guided Demo 11 – Scheduling & Monitoring 175
Guided Demo 12 – Detailed Logging throughout Execution 179
Appendix – Course Slides 188

Guided Demo 1 – Launch & Customize PDI

Introduction

In this guided demo, you launch Spoon, PDI's graphical designer. Then, you learn how to customize some of its options and default behavior.

Objectives

In this guided demo, you will:

- Launch Spoon
- Open Spoon's 'Options' dialog
- Describe the common options and look & feel settings
- Toggle the welcome page
- Toggle the repository's dialog at startup
- Change the grid settings

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured.

Guided Demo 1 – Launch & Customize PDI, Continued

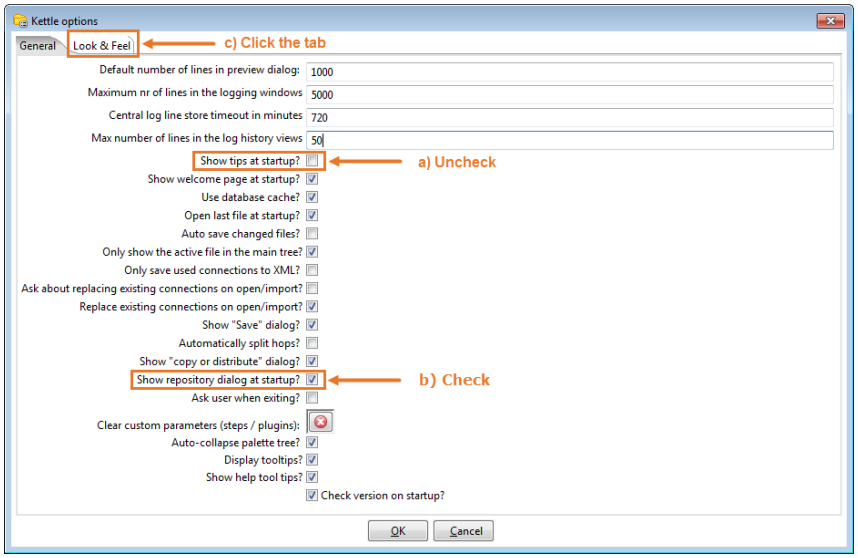
Guided Demo In this section of the guided demo, you launch Spoon.

Step	Action
1	To launch Spoon, PDI's graphical designer interface, from the Windows Start button: **Start > All Programs > Pentaho Enterprise Edition > Design Tools > Data Integration** 👍 TIP: Create a shortcut to Spoon on the Desktop to easily start the interface throughout this course.
2	Read the Tip, and then, click the **[Close]** button. The PDI application with the 'Welcome!' screen is displayed.
3	**Scroll** the 'Welcome!' screen to familiarize yourself with its contents. 
4	In the menu bar, click **Tools** | **Options…**

Continued on next page

Guided Demo 1 – Launch & Customize PDI, Continued

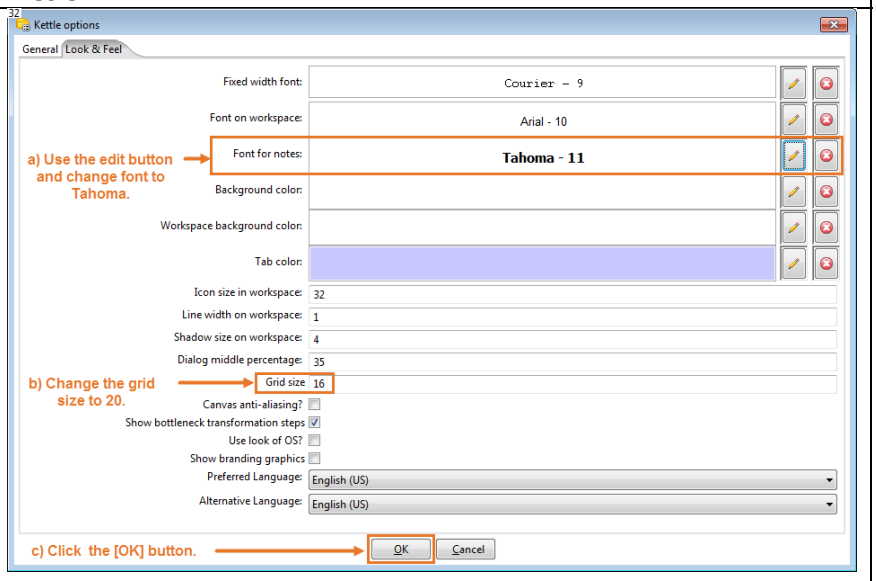
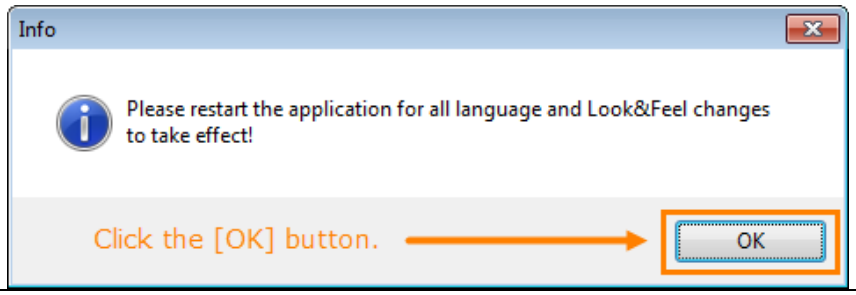
Guided Demo, continued

Step	Action
5	■ Uncheck the 'Show tips at startup?' checkbox. ■ Check the 'Show repository dialog at startup?' checkbox. ■ Click the 'Look & Feel' tab.  🕮 NOTE: We will explain repositories in future models. It is in those modules where you begin saving objects and configurations to a database using the repository.
6	■ Click the **edit** button  for the 'Font for notes' property. The 'Font' dialog is displayed, change the font to '**Tahoma**' and then click the **[OK]** button. ■ Change the '**Grid size**' property to **16.** ■ Click the **[OK]** button to accept your changes and close the 'Kettle options' dialog. The 'Info' dialog will display.

Continued on next page

Guided Demo 1 – Launch & Customize PDI, Continued

Guided Demo, continued

Step	Action
6 (cont)	 📖 NOTE: Feel free to use any font or grid size you like. However, 32 is the recommended setting for the grid size because it makes aligning steps on the canvas easy.
7	Click the **[OK]** button to close the 'Info' dialog, and then, **close Spoon**. 

Continued on next page

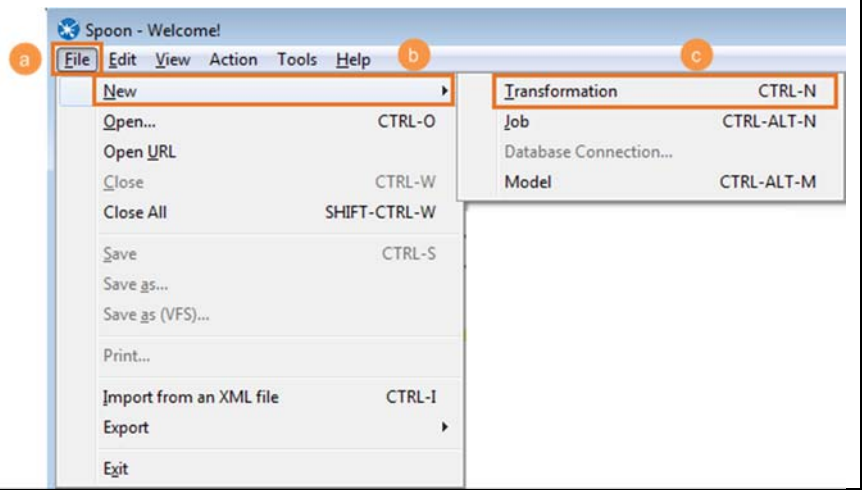
Guided Demo 1 – Launch & Customize PDI, Continued

NOTE

Although restarting Spoon after changing Look & Feel options is the best habit to have, not all options require a restart for the changes to take effect.

Verify User Interface Changes

In this section of the guided demo, you will verify the user interface and other changes to the Spoon options took effect.

Step	Action
1	Launch Spoon. Notice the following: ■ The 'Repository Connection' dialog is displayed. Click **[Cancel]**. ■ The 'Spoon tips…' dialog is not displayed.
2	To create a new Transformation, in the menubar, click **File | New | Transformation** 

Continued on next page

Guided Demo 1 – Launch & Customize PDI, Continued

Verify User Interface Changes, continued

Step	Action
3	**Right-click** anywhere on the empty canvas, and then click **New note** in the context menu that appears.
4	Click in the 'Note' section of the 'Notes' dialog, type an informational message about your transformation and then click the **[OK]** button. A new note with the text you entered is displayed on the canvas.

TIP

Turn the Spoon tips back on so that you can read helpful tips every time you start Spoon in this course. Changing the grid size to 32 will also allow for easy step alignment to the canvas as you create transformations.

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Guided Demo 2 – Creating a 'Hello World' Transformation

Introduction

In this guided demo, we will use Spoon to create a new transformation with data containing "Hello World". Although this is probably not a task you will be asked to do in the real world, the concepts learned in this guided demo help to build the foundation necessary for creating any transformation.

Objectives

In this guided demonstration you will…

- Learn to create a new transformation.
- Add steps and hops.
- Configure the Generate rows step.

Creating the Transformation the Steps, and Executing

Step	Action
1	Launch Spoon.
2	To create a new transformation, in the toolbar, click the '**New file**'  icon, and then click **Transformation**.
3	To create a new step in the transformation: ■ In the 'Design and View Tabs' panel on the left, click the **Design** tab. A tree of step categories is displayed. ■ Expand the **Input** node. ■ Drag-and-drop the **Generate Rows** step from the Input step category to the canvas. A new step is created in the transformation.
4	In the canvas, double-click on the **Generate Rows** step. The 'Generate Rows' dialog is displayed.

Continued on next page

Guided Demo 2 – Creating a 'Hello World' Transformation, Continued

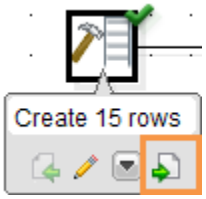
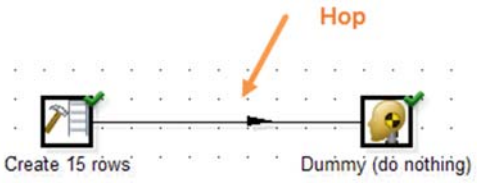
Creating the Transformation the Steps, and Executing, continued

Step	Action
5	Enter values in the properties of the 'Generate Rows' dialog as follows: **Property Name** / **Value** Step name: Create 15 rows Limit: 15
6	Fill in the 'Fields' grid if the 'Generate Rows' dialog as follows: **Column Name** / **Value** Name: Greeting Message Type: String Value: Hello, World! IMPORTANT: Use exact spelling and case if you choose to type values for columns in a grid that have drop-down lists.
7	Before we close this dialog and continue creating the transformation, let's make certain the step generates the data we expect. ■ Click on the **[Preview]** button. The 'Enter preview size' dialog is displayed. ■ In the 'Enter preview size' dialog, click the **[OK]** button. ■ Verify 15 rows of data with the message you entered is displayed, and then click the **[OK]** button to close the 'Examine preview data' dialog. ■ Click the **[OK]** button to close the 'Generate Rows' dialog. TIP: Previewing data and testing steps along the way can really help to minimize errors and trouble-shooting time later in the transformation creation process.
8	In the 'Design and View Tabs' panel, expand the **Flow** branch, and then drag-and-drop the **Dummy (do nothing)** step to the canvas, to the right of the existing step.

Continued on next page

Guided Demo 2 – Creating a 'Hello World' Transformation, Continued

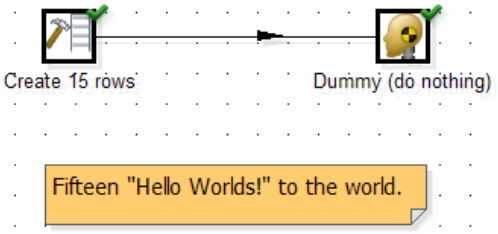
Creating the Transformation the Steps, and Executing, continued

Step	Action
9	Now you will create a hop between the two steps. ■ Hover the mouse pointer over the Generate Rows step. A small toolbar will appear under the step. ■ Click on the output connector icon.  ■ Click on the **Dummy (do nothing)** step. A hop is created between the two steps.  TIP: Another method to create a hop between steps is: ■ Click and hold on the source step using the middle mouse button. ■ Point to the destination step. ■ Release the middle mouse button.
10	**Right-click** anywhere on the empty canvas, and then click '**New note**' in the context menu that appears.

Continued on next page

Guided Demo 2 – Creating a 'Hello World' Transformation, Continued

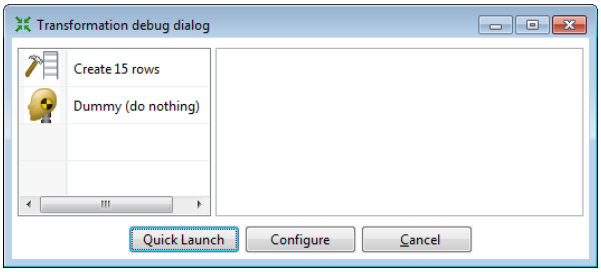
Creating the Transformation the Steps, and Executing, continued

Step	Action
11	Type an informational message about your transformation and then click the **[OK]** button. A new note with the text you entered is displayed on the canvas. 
12	To set the transformation properties: ■ In the menu bar, click **Edit | Settings**. ■ Provide a **name** for the transformation, such as 'HelloWorld'. ■ Optionally, enter a more detailed description in the 'Extended description' property. ■ Click the **[OK]** button.
13	Use the **File | Save** from the menu, save icon in the toolbar, or press **CTRL-S** to save the transformation.
14	Use the mouse to drag a box that surrounds both steps and the hop. Both steps will be selected.

Continued on next page

Guided Demo 2 – Creating a 'Hello World' Transformation, Continued

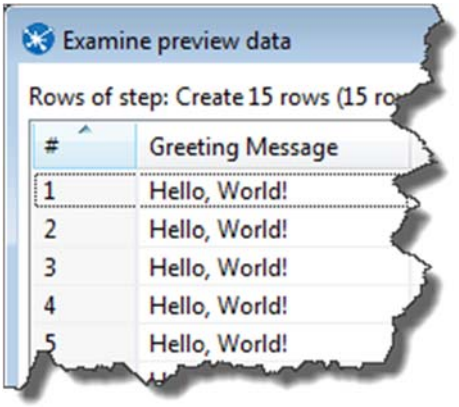
Creating the Transformation the Steps, and Executing, continued

Step	Action
15	Click the Preview icon in the Sub-toolbar, located immediately above the canvas. The 'Transformation debug dialog' dialog is displayed. 

Continued on next page

Guided Demo 2 – Creating a 'Hello World' Transformation, Continued

Creating the Transformation the Steps, and Executing, continued

Step	Action
16	Click the **[Quick Launch],** and then click **[Show]**. The 'Examine preview data' dialog is displayed.  This dialog is the same dialog that was displayed when previewing the data from the Generate Rows step. This is because the Dummy (do nothing) step does not perform any logic, so the same data is expected.
17	Click the **[Close]** button to close the 'Examine preview data' dialog.

Solution Details

The solution to this guided demonstration can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Guided Demonstrations`

Completed transformation:
`GD2_HelloWorld.ktr`
(Use **File | Import from an XML file** to import)

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Exercise 1 – Generate Rows, Sequence, Select Values

Create the transformation

In the first part of this exercise, you will create a transformation, and then separately add and configure each step and connecting hops. You will preview each step after it is added and configured to ensure the data previewed is what is expected. This best practice technique helps to prevent configuration errors that would otherwise be difficult to troubleshoot if the entire transformation was created, and configured prior to preview.

Please concentrate on learning the techniques for creating the transformation, rather than the logic of the steps being used. You will learn the details of these and many other steps throughout the remainder of this course.

Objectives

In this exercise you will…

- Learn to create a new transformation.
- Add steps and hops.
- Configure and preview steps
- Execute the transformation.

Create the Transformation and add the Generate Rows step

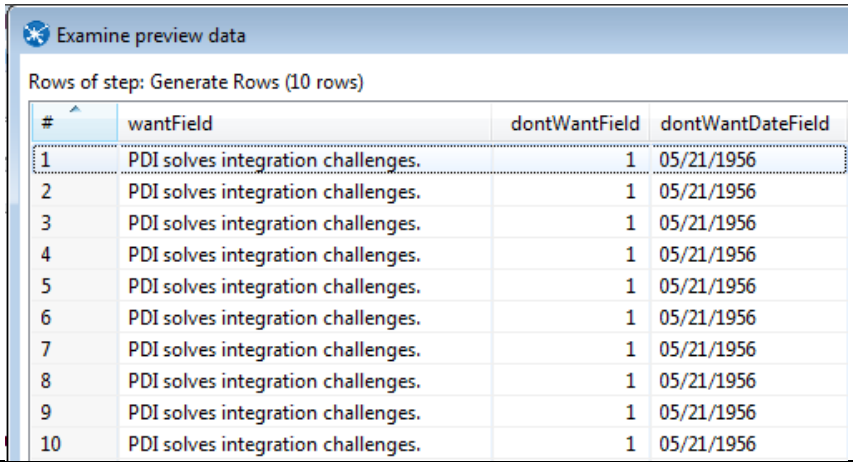
To create the transformation:

Step	Action
1	In Spoon, click **File** | **New** | **Transformation**.
2	To add the first step, expand the **Input** category in the **Design tab**, and then drag the **Generate Rows** step onto the canvas.
3	To open the step properties dialog, in the canvas, **double-click** the **Generate rows** step.

Continued on next page

Exercise 1 – Generate Rows, Sequence, Select Values, Continued

Create the Transformation and add the Generate Rows step, continued

Step	Action
4	To configure the 'Generate rows' step, set the step's properties as shown in the table below: **Property Name** / **Value** Step name: Generate 10 Rows Limit: 10
5	To configure the Fields grid, add three rows configured as shown in the table below: **Name** / **Type** / **Format** / **Value** wantField / String / / PDI solves integration challenges. dontWantField / Integer / # / 1 dontWantDateField / Date / MM/dd/yyyy / 05/21/1956 📖 NOTE: You must use the exact case and spelling as shown in the table above.
6	To preview the data and confirm it is configured properly, for the 'Generate Rows', click **[Preview]**, and then click **[OK]**.
7	Verify the data generated is correct by comparing your data with the screenshot shown below:  Examine preview data Rows of step: Generate Rows (10 rows) # / wantField / dontWantField / dontWantDateField 1 PDI solves integration challenges. 1 05/21/1956 2 PDI solves integration challenges. 1 05/21/1956 3 PDI solves integration challenges. 1 05/21/1956 4 PDI solves integration challenges. 1 05/21/1956 5 PDI solves integration challenges. 1 05/21/1956 6 PDI solves integration challenges. 1 05/21/1956 7 PDI solves integration challenges. 1 05/21/1956 8 PDI solves integration challenges. 1 05/21/1956 9 PDI solves integration challenges. 1 05/21/1956 10 PDI solves integration challenges. 1 05/21/1956
8	To close the 'Examine preview data' dialog, click **[OK]**.

Continued on next page

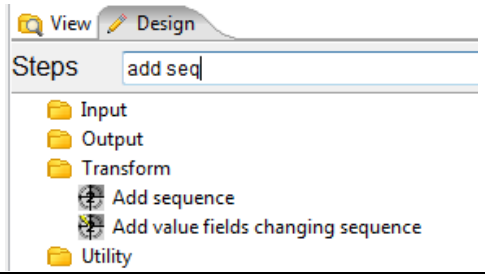
Exercise 1 – Generate Rows, Sequence, Select Values, Continued

Create the Transformation and add the Generate Rows step, continued

Step	Action
9	To close the 'Generate Rows' dialog, click **[OK]**.
10	To save the transformation, in the Spoon menu, click **File | Save**, and then provide transformation details as shown in the table below: **Property Name** / **Value** Transformation name: SelectValues Description: (enter a description of your choosing) Extended description: (enter an extended description of your choosing) Directory: C:\pentahotraining\My Work\EX1
11	To close the Transformation properties dialog and save the transformation, click **[OK]**.

Create & Configure the Add Sequence Step

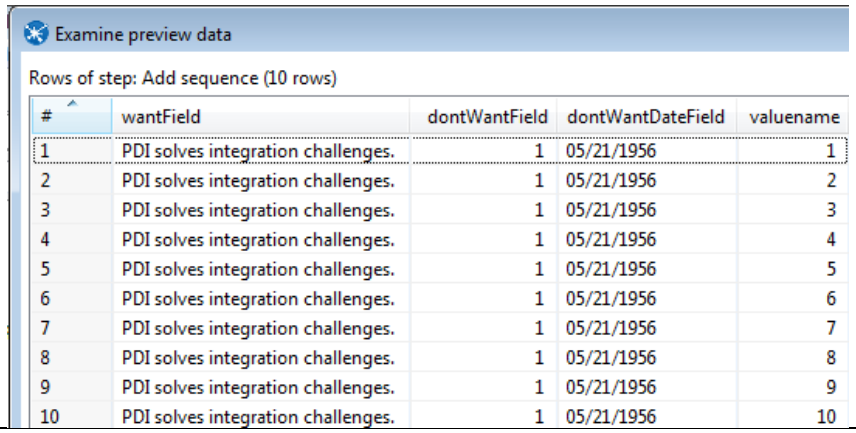
To add and configure the 'Add sequence' step:

Step	Action
1	To find the second step to add to the transformation, in the **Design** tab, type '**add seq**' in the search field. The Design tab's tree will automatically be filtered to display the steps that match the text entered as shown in the screenshot below: 
2	To add the step, **drag** the **Add sequence** step to the canvas, placing it to the right of the first step.

Continued on next page

Exercise 1 – Generate Rows, Sequence, Select Values, Continued

Create & Configure the Add Sequence Step, continued

Step	Action
3	Add a hop from the first step, to the second step.
4	The 'Add sequence' step's default configuration is exactly how we want ours configured, therefore, no configuration is necessary. To examine its configuration, **double-click the step**, notice its default configuration, and then click **[OK]**.
5	To preview the Add Sequence step: ■ Select the step. ■ In the Spoon menu, click **Action** | **Preview**. ■ Click the **[Quick Launch]** button.
6	Verify the preview data has all of the first step's fields as well as the new field from the 'Add sequence' step, named 'valuename'. 
7	To close the 'Examine preview data' dialog, click **[Close]**.
8	Save the transformation.

#	wantField	dontWantField	dontWantDateField	valuename
1	PDI solves integration challenges.	1	05/21/1956	1
2	PDI solves integration challenges.	1	05/21/1956	2
3	PDI solves integration challenges.	1	05/21/1956	3
4	PDI solves integration challenges.	1	05/21/1956	4
5	PDI solves integration challenges.	1	05/21/1956	5
6	PDI solves integration challenges.	1	05/21/1956	6
7	PDI solves integration challenges.	1	05/21/1956	7
8	PDI solves integration challenges.	1	05/21/1956	8
9	PDI solves integration challenges.	1	05/21/1956	9
10	PDI solves integration challenges.	1	05/21/1956	10

Continued on next page

Exercise 1 – Generate Rows, Sequence, Select Values, Continued

Create & Configure the Select Values Step

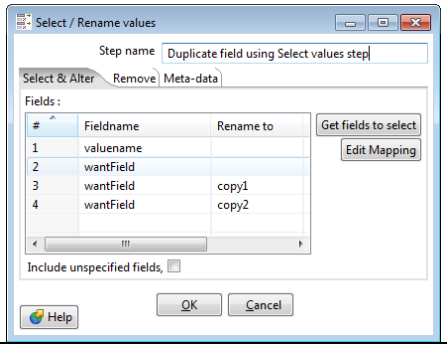
In this section of the exercise, you will add a 'Select values' step that will show you how to copy a field in the stream, including its data, and add it to the stream as a new field. You will duplicate the 'wantField' field twice.

Step	Action
1	To create the third step, from the **Transform** category of the **Design** tab, drag the **Select values** step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step** | **Destination Step** Add sequence | Select values
3	To open the step's properties dialog, **double-click the step**.
4	To configure the step's properties, set them according to the table shown below: **Property Name** | **Value** Step Name | Duplicate field using Select values step
5	To configure the fields to select and duplicate to new fields, in the **Select & Alter** tab, set the grid according to the table below: **Fieldname** | **Rename to** valuename | <empty> wantField | <empty> wantField | copy1 wantField | copy2

Continued on next page

Exercise 1 – Generate Rows, Sequence, Select Values, Continued

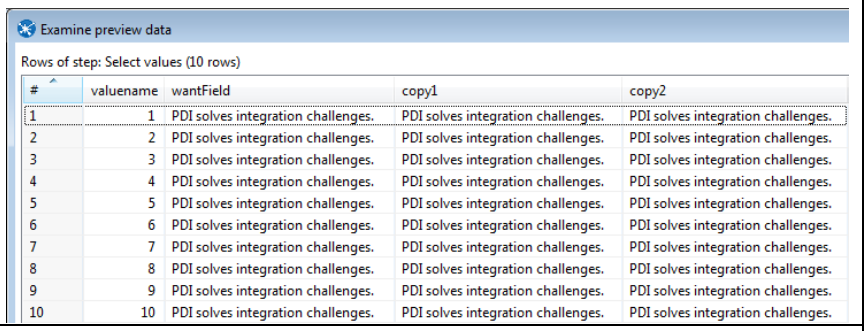
Create & Configure the Select Values Step, continued

Step	Action
6	Compare the step's configuration with the screenshot below and make any necessary changes. 
7	To close the step's properties dialog, click **[OK]**.

Continued on next page

Exercise 1 – Generate Rows, Sequence, Select Values, Continued

Create & Configure the Select Values Step, continued

Step	Action
8	To preview the Select values step: ■ Select the step. ■ In the Spoon menu, click **Action | Preview**. ■ Click the **[Quick Launch]** button.
9	Verify the preview data has all of the first step's fields as well as the new field from the Add Sequence step, named valuename. 
10	To close the 'Examine preview data' dialog, click **[Close]**.
11	Save the transformation.

#	valuename	wantField	copy1	copy2
1	1	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
2	2	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
3	3	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
4	4	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
5	5	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
6	6	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
7	7	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
8	8	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
9	9	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.
10	10	PDI solves integration challenges.	PDI solves integration challenges.	PDI solves integration challenges.

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Exercises`

Completed transformation:
`EX1_SelectValues.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Guided Demo 3 – Error Handling & Basic Logging

Introduction

This guided demonstration provides and introduction to finding and handling errors, as well as logging.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Objectives

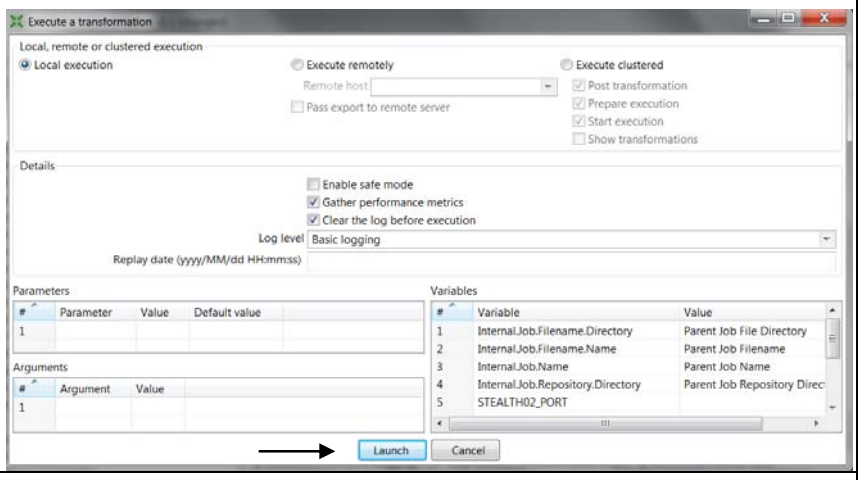
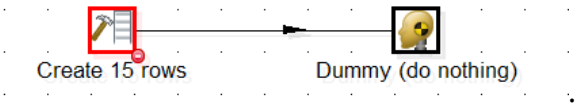
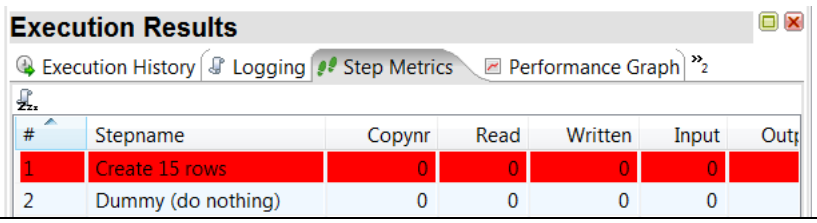
After completing this guided demonstration, you will be able to:

- Know when you have an error.
- Find the log file.
- Show only Error messages.

Continued on next page

Guided Demo 3 – Error Handling & Basic Logging, Continued

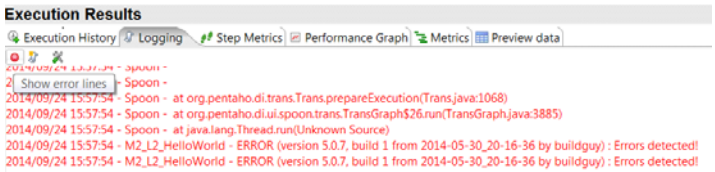
Error Handling and Logging

Step	Action
1	If it is not already, **open** the **HelloWorld** transformation.
2	**Double-click** on the Generate rows step named, 'Create 15 rows'.
3	Change the 'Greeting Message' from String to Integer. This will cause the transformaion to error. Click **[OK]**.
4	Run the transformation by clicking on the green arrow in the sub-toolbar. The icon that is the farthest to the left.
5	The launch dialogue will appear, click the launch button. 
6	On the canvas you will see that the 'Create 15 rows' step is now outlined in red. If you click on this icon it will show you the errors. 
7	Notice that in the 'Step Metrics' tab there is a large red line through the step that has the error. 

Guided Demo 3 – Error Handling & Basic Logging, Continued

Error Handling and Logging

continued

Step	Action
8	The best way to see the error is to look in the log file. To do this click on the Logging tab. Error lines are in red. 
9	On the logging tab you will see and icon on the far left. Click on this icon, a dialogue box will appear that will show you ONLY the error lines. This makes it much easier to find your errors. 
10	Return to the Create 15 rows step and change the field type back to String.

Continued on next page

Guided Demo 3 – Error Handling & Basic Logging, Continued

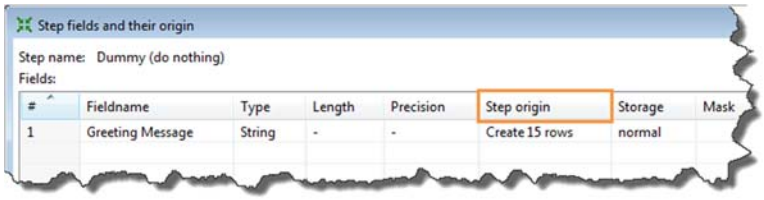
Error Handling & Logging, continued

Step	Action
11	Rerun the transformation.

TIP

When troubleshooting a transformation, it can be very helpful to know exactly where fields on a step originate from. To do this:

Right-click the step you wish to learn more about, and then select **Show input fields**, or **Show output fields**. You will be presented with a grid that shows the step each field originated from.



Solution Details

The solution to this guided demonstration can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Guided Demos`

Completed transformation:
`GD3_HelloWorldWithError.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this guided demonstration.

Guided Demo 4 – Saving a Transformation in the Repository

Introduction

In this guided demonstration, you will get a brief introduction to the repository and learn how to save an existing transformation into it. You can then use the repository throughout this course to save and organize your transformations and jobs.

Objectives

After completing this guided demonstration, you will be able to:

- Connect to an existing repository.
- Save transformations and jobs into an existing repository.

Prerequisites

Access to a student environment where a repository has already been created for you.

Repositories

Repositories are objects that provide a location to save transformations, jobs, and their configurations (metadata). PDI allows users to create and use three types of repositories, which will be looked at in detail later in this course. By default, no repository is used and transformations and jobs are stored on the local file system as individual *.ktr and *kjb files in XML format.

📖 **NOTE**

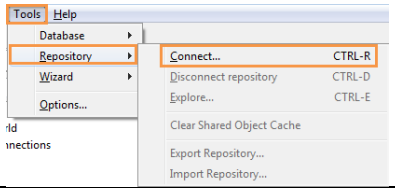
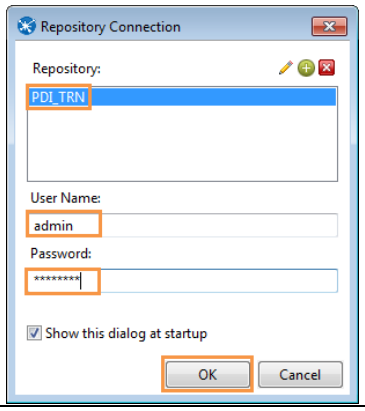
The Pentaho hosted student environments already have a repository created.

Continued on next page

Guided Demo 4 – Saving a Transformation in the Repository, Continued

Open a Recent Transformation and Connect to a Repository

In this section of the guided demonstration, you will open a recent transformation and connect to an existing repository.

Step	Action
1	To open a recent transformation, in the Spoon menubar, click **File | Open Recent**, and then, click the **HelloWorld** transformation.
2	To open the 'Repository connection' dialog, in the Spoon menubar, click **Tools | Repository | Connect…** 
3	To connect to the PDI_TRN repository: ■ In the 'Repository connection' dialog, select **PDI_TRN**. ■ In the User Name field, enter **'admin'**. ■ In the password field, enter '**password**'. ■ Click **[OK]**. 

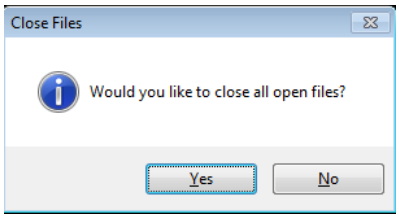
CAUTION The fields are case sensitive. The User Name field will likely already be populated for you.

Continued on next page

Guided Demo 4 – Saving a Transformation in the Repository, Continued

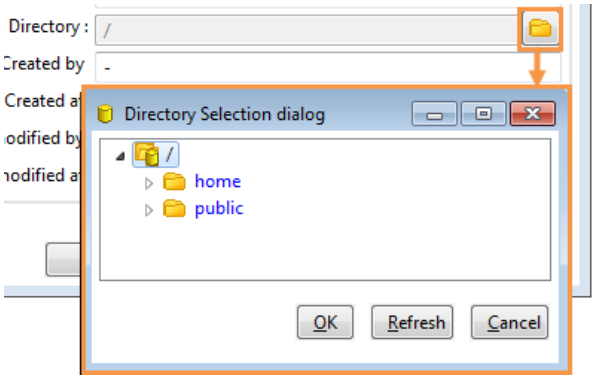
The Close Files Dialog

If you receive a dialog as shown below, and you want to keep your existing files open when Spoon starts, click **[No]**.



Transformation Properties Dialog when Connected to a Repository

When you *are* connected to a repository, the transformation (or Job) properties dialog allows you to save the object to the repository by clicking on the folder icon for the Directory property. This is illustrated in the screenshot below:



When you are *not* connected to a repository, the Transformation (or Job) properties dialog only allows viewing where the object is currently saved (if it has been saved previously).

TIP

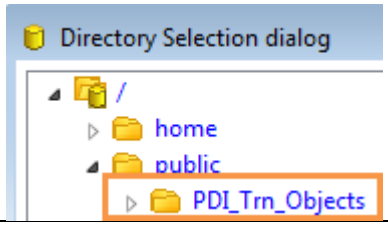
Even while you are connected to the repository, you can save an object to your environment's file system by using the File | Save As… options in the Spoon menubar.

Continued on next page

Guided Demo 4 – Saving a Transformation in the Repository, Continued

Save the Transformation in the Repository

This final portion of the guided demonstration has you save the open transformation into the repository that you are now connected to.

Step	Action
1	To open the Transformation properties dialog, press **CTRL-T**, or double-click an empty area on the Canvas.
2	To save the transformation to the repository: ■ In the Transformation dialog, click the folder icon for the Directory property. ■ In the 'Directory Selection dialog', right-click '**public'**, and then click **'New sub-directory'**. Name it '**PDI_Trn_Objects**'. The new folder will be selected. ■ Click **[OK]**. 
3	Save the transformation.
4	To verify the transformation is saved in the repository: ■ Press **CTRL-T** to open the Transformation properties dialog. ■ Examine the **Directory** property and see it has the repository folder structure as you defined.
5	To close the Transformation dialog, click **[OK]**.

📖 **NOTE**

Throughout this course, please feel free to create new folders and organize your transformations and jobs in the repository or file system as you like. The save locations and names in this guide are only suggestions.

End of Guided Demonstration

Congratulations! You have completed this guided demonstration. Now you can use the repository to save your work throughout this course.

Guided Demo 5 – Combining Several Inputs into One Output

Introduction

In this guided demo, you will create a transformation that will read multiple text files using a regular expression. Then, it will add the system date/time and transformation modified date/time to the stream. Finally, the entire stream will output to one delimited text file (CSV).

Prerequisites

To complete this exercise, you need access to the input files that reside on the student environment for this course.

Objectives

After completing this guided demo, you will be able to:

- Configure a 'Text file input' step to read multiple text files based on a regular expression.
- Exclude a specific filename to prevent it from being included as an input.
- Add various system and environment related data to the stream.
- Create a single output file that is delimited (CSV).

Continued on next page

Guided Demo 5 – Combining Several Inputs into One Output, Continued

Create the Transformation

Step	Action
1	To create the new transformation, in the Spoon menubar, click **File** | **New** | Transformation.
2	To open the Transformation properties dialog, in the Canvas, double-click on an empty area.
3	Set the transformation properties for the **Transformation tab** according to the table below:

Property Name	Value
Transformation name	MultiInputOneOutput
Description	Add a description of your choice.
Directory	/public/PDI_Trn_Objects 🕮 NOTE: This is in the repository.

Step	Action
4	To close the Transformation dialog, click **[OK]**.

Create & Configure the Text File Input Step

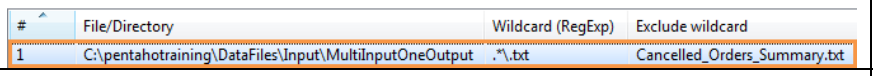
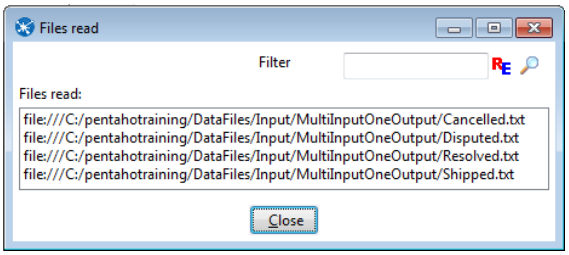
The first step in this transformation will read multiple text files from an input directory. It will also use regular expressions to determine which files to include and which one to exclude.

Step	Action
1	To create the first step, from the **Input** category of the **Design** tab, drag the **Text file input** step onto the Canvas.
2	To open the step's properties dialog, double-click the step.

Continued on next page

Guided Demo 5 – Combining Several Inputs into One Output, Continued

Create & Configure the Text File Input Step, continued

Step	Action
3	To configure the step's properties, set them according to the table shown below: **Property Name** / **Value** Step Name: Text File Input from Multiple Files File or directory: C:\pentahotraining\DataFiles\Input\MultiInputOneOutput IMPORTANT: Type in the path. Do not use the Browse button. Browse is used to point to a specific file. Regular Expression: .*\.txt Exclude Regular Expression: Cancelled_Orders_Summary.txt
4	To add the file and expression configuration to the Selected files grid, click **[Add]**. It will be added to the grid as shown in the screenshot below. 
5	Verify the correct files will be read and the expressions entered are correct by clicking the **[Get filenames…]** button.  🕮 NOTE: Notice how the Cancelled_Orders_Summary.txt file is not listed. The exclusion expression prevents it from being included.

Continued on next page

Guided Demo 5 – Combining Several Inputs into One Output, Continued

Create & Configure the Text File Input Step, continued

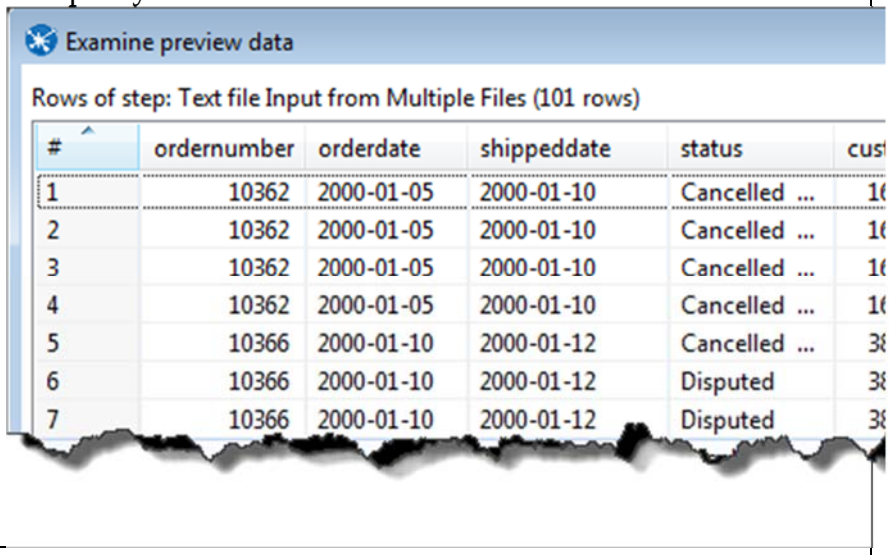
Step	Action
6	To close the 'Files read' dialog, click **[Close]**.
7	To configure the fields for this step: ■ Click the **Fields** tab. ■ Click the **[Get Fields]** button. ■ At the 'Sample size' dialog, click **[OK]**. ■ At the 'Scan results' dialog, click **[Close]**.
8	Verify your fields to the screenshot shown below: 📖 NOTE: The 'Text file input' step must always be configured to read all of the fields in the file. You cannot choose a subset of the fields in the file and have it successfully read the data.

#	Name	Type	Format	Position	Length
1	ordernumber	Integer	#		15
2	orderdate	Date	yyyy-MM-dd		
3	shippeddate	Date	yyyy-MM-dd		
4	status	String			15
5	customernumber	Integer	#		15
6	orderlinenumber	Integer	#		15
7	productcode	String			15
8	quantityordered	Integer	#		15
9	priceeach	Number	#,#		15

Continued on next page

Guided Demo 5 – Combining Several Inputs into One Output, Continued

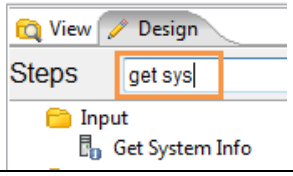
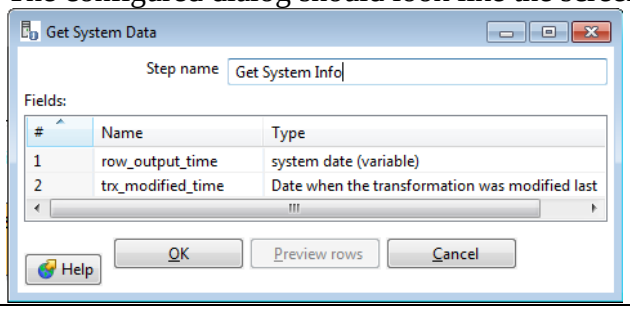
Create & Configure the Text File Input Step, continued

Step	Action
9	To verify the step is configured properly, preview the data by clicking **[Preview]**, and then, click **[OK]**. Compare your data with the same shown: 
10	To close the 'Examine preview data' dialog, click **[Close]**.
11	To close the step's properties dialog, click **[OK]**.
12	Save the transformation.

Continued on next page

Guided Demo 5 – Combining Several Inputs into One Output, Continued

Create & Configure the Get System Info Step

Step	Action
1	To use the search box and find the next step to add to your transformation, from the **Design** tab, enter the following text into the search box: "**get sys**" 
2	To create the first step, from the **Input** category of the **Design** tab, drag the **Get System Info** step onto the Canvas.
3	Create a hop between the steps as shown in the table below: **Source Step**: Text file input **Destination Step**: Get System Info
4	To open the step's properties dialog, double-click the step.
5	To configure the **Fields** grid, add two rows according to the table below: **Name**: row_output_time; **Type**: System date (variable) **Name**: trx_modified_time; **Type**: Date when the transformation was modified last The configured dialog should look like the screenshot below: 
6	To close the step's properties dialog, click **[OK]**.
7	Save the transformation.

Continued on next page

Guided Demo 5 – Combining Several Inputs into One Output, Continued

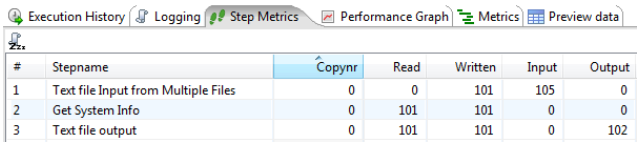
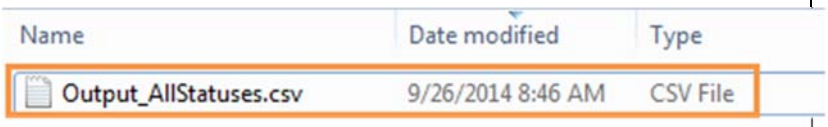
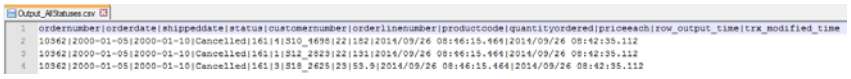
Create & Configure the Text File Output Step

Step	Action
1	To create the first step, from the **Output** category of the **Design** tab, drag the **Text file output** step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step** → **Destination Step** Get System Info → Text file output
3	To open the step's properties dialog, double-click the step.
4	To configure the Filename property, in the **File** tab, set it according to the table shown below: **Property Name**: Filename **Value**: C:\pentahotraining\DataFiles\Output\Output_AllStatuses 📖 NOTE: Do not add a file extension in the Filename property. There is another step Extension property allows you to set the extension.
5	To configure the fields to include in the output file: ■ Click the **Fields** tab. ■ Click the **[Get Fields]** button.
7	To close the step's properties dialog, click **[OK]**.
8	Save the transformation.

Continued on next page

Guided Demo 5 – Combining Several Inputs into One Output, Continued

Execute the Transformation

Step	Action
1	To run the transformation, press **F9**, and then, click **[Launch]**.
2	There should be no errors and the **Step Metrics** tab in the Execution Results pane should look similar to the screenshot shown below:  Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data \# | Stepname | Copynr | Read | Written | Input | Output 1 | Text file Input from Multiple Files | 0 | 0 | 101 | 105 | 0 2 | Get System Info | 0 | 101 | 101 | 0 | 0 3 | Text file output | 0 | 101 | 101 | 0 | 102 🕮 NOTE: Notice the Output field contains data. This is indicates the number of lines output to the text files that it created.
3	To verify the text file was created, navigate to the folder shown below and verify the creation of the file as show in in the example screenshot: C:\pentahotraining\DataFiles\Output  Name | Date modified | Type Output_AllStatuses.csv | 9/26/2014 8:46 AM | CSV File
4	Open the text files and notice how orders from all of the input files are included. Example of **Output_AllStatuses.csv**  Output_AllStatuses.csv ordernumber|orderdate|shippeddate|status|customernumber|orderlinenumber|productcode|quantityordered|priceeach|row_output_time|trx_modified_time 10362|2000-01-05|2000-01-10|Cancelled|161|4|S10_4698|22|152|2014/09/26 08:46:15.466|2014/09/26 08:42:35.112 10362|2000-01-05|2000-01-10|Cancelled|161|1|S12_2823|22|131|2014/09/26 08:46:15.466|2014/09/26 08:42:35.112 10362|2000-01-05|2000-01-10|Cancelled|161|3|S18_2625|25|53.9|2014/09/26 08:46:15.466|2014/09/26 08:42:35.112 🕮 NOTE: In the example there are no multiple trailing spaces at the end of the values. This is because the Minimum width button was used in the Text file output step's Fields tab.

Continued on next page

Guided Demo 5 – Combining Several Inputs into One Output, Continued

Solution Details

The solution to this guided demonstration can be obtained using the details below:

Location: `C:\pentahotraining\Solutions\Guided Demos`

Completed transformation:

`GD5_MultiInputOneOutput.ktr`
(Use **File | Import from an XML file** to import)

Text file output: `Output_AllStatuses.txt`

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Guided Demo 6 – Creating kettle.properties Variables

Introduction

It's important in PDI to create transformations that will run in a variety of environment. That can easily be ported from Development to Test to Production. This guided demo shows how to create flexible transformations by using 'kettle.properties' variables.

Prerequisites

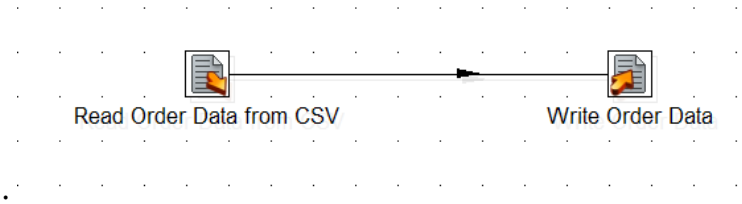
You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Objectives

After completing this guided demonstration, you will be able to:

- Find and edit the kettle.properties file.
- Create 2 new variables DIR_INPUT and DIR_OUTPUT.
- Use these new variables in a transformation to read and write files.

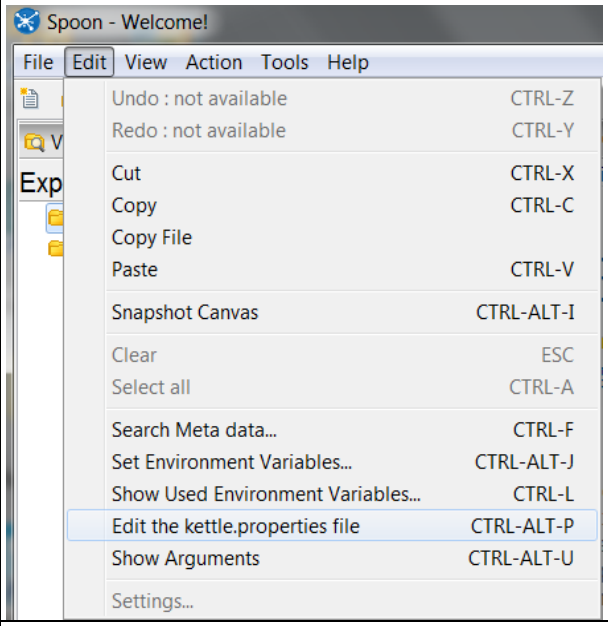
Model



Continued on next page

Guided Demo 6 – Creating kettle.properties Variables, Continued

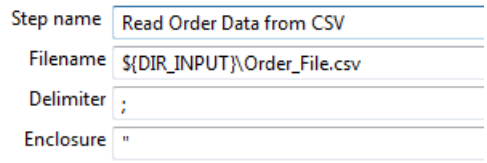
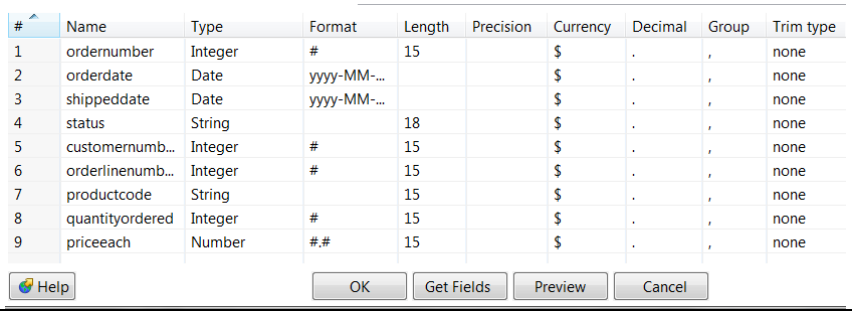
Creating Variables for File Paths

Step	Action
1	Create a new transformation
2	Save the transformation as **KettleVariables**.
3	To edit the kettle.properties file, in the menu, click on **Edit |Edit the kettle.properties file**. 
4	The kettle.properties file will open, scroll to the bottom of file and right click on the very last line. A list of grid options will appear. Choose the second item on the list **Insert after this row**. A blank line will appear. Create **two** blank lines.
5	In the empty lines enter these Variables **Variable name** / **Value** DIR_INPUT / C:\pentahotraining\DataFiles\Input DIR_OUTPUT / C:\pentahotraining\DataFiles\Output
6	To close the kettle.properties file, click **[OK]**.
7	Drag the CSV file input to the canvas.

Continued on next page

Guided Demo 6 – Creating kettle.properties Variables, Continued

Creating Variables for File Paths

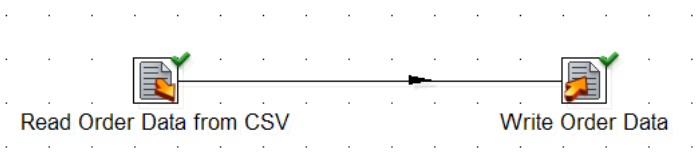
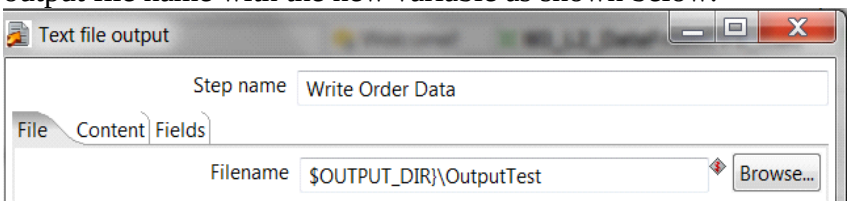
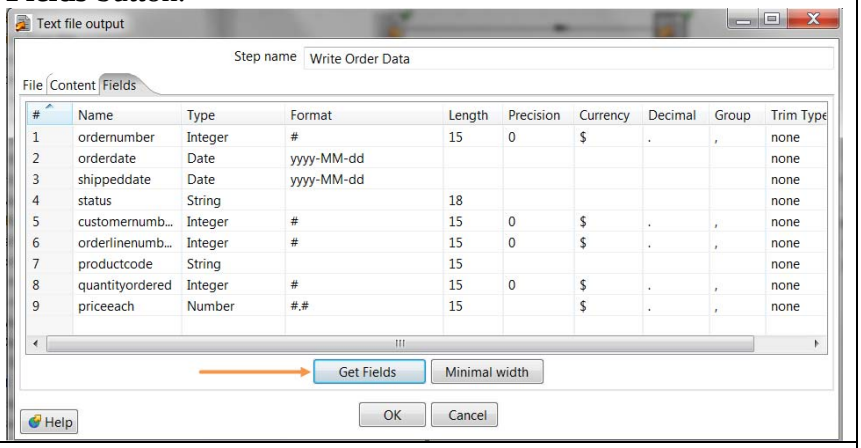
Step	Action
8	Double click on the step to open it. In the **Step Name** field type Read Order Data from CSV.
9	Fill in the step as shown below using the new variable DIR_INPUT.  🕮 NOTE: Parameters can be used in any field in Spoon that has a red and gray diamond. Using CTRL-Space in the field will open a list of the current parameters.
10	Make sure that the **Separator** is set to semicolon (;). Enable **Header** because there is one line of header rows in the file.
11	Click the GetFields tab and click Get Fields to retrieve the input fields from your source file. A dialogue box will pop-up asking for a sample size (number of rows). Enter 300. 
12	Click **Preview** to verify that your file is being read correctly. You can change the number of rows to preview. Click **[OK]** to exit the step properties dialog box.
13	Open the Output Category and drag a Text File Output step to your transformation.

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	ordernumber	Integer	#	15		$	.	,	none
2	orderdate	Date	yyyy-MM-...			$	.	,	none
3	shippeddate	Date	yyyy-MM-...			$	.	,	none
4	status	String		18		$	.	,	none
5	customernumb...	Integer	#	15		$	.	,	none
6	orderlinenumb...	Integer	#	15		$	.	,	none
7	productcode	String		15		$	.	,	none
8	quantityordered	Integer	#	15		$	.	,	none
9	priceeach	Number	#.#	15		$	.	,	none

Continued on next page

Guided Demo 6 – Creating kettle.properties Variables, Continued

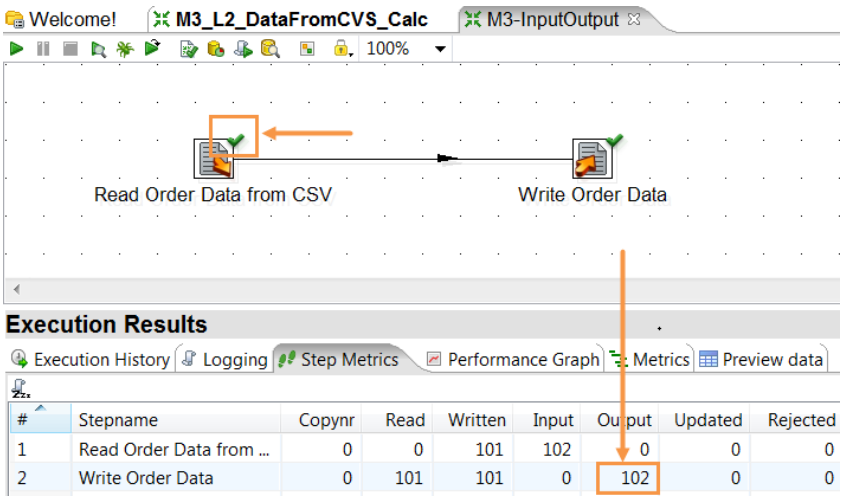
Creating Variables for File Paths

Step	Action
14	Create the Hop between the two steps 
15	Open the Text File Output step In the **Step Name** field type Write Order Data
16	Open the Text File Output step, in the **File** tab and fill in the output file name with the new variable as shown below.  .
17	Leave the **Content** tab as is. Note the delimiter will be a semicolon.
18	Click on the **Fields** tab on the bottom of the page click the **Get Fields** button. 
19	Click **[OK]** to close the dialog.
20	**Run** the transformation by clicking on the icon.

Continued on next page

Guided Demo 6 – Creating kettle.properties Variables, Continued

Model, continued

Step	Action
21	Notice the green check marks on the steps, this means the transformation has run successfully. You can see on the **Step Metrics** tab that a file was created. 
22	Check to see if the OutputTest.txt file has been created at: **C:\pentahotraining\DataFiles\Output**

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Guided Demos`

Completed transformation:

`GD6_KettleParameters.ktr`
(Use **File | Import from an XML file** to import)

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case

Introduction

In this exercise, you will create a transformation that reads a CSV file containing order and country of origin data. Then, it will send incoming data for specific country's orders to text files that it creates. Previously, you used kettle.properties parameters. Here, you will use transformation parameters; giving you experience using both types.

📖 NOTE

For those looking for more of a challenge, try the advanced version of this exercise. It is the same exercise, but without the detailed guidance. You will find it in this workbook immediately following this exercise.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Objectives

After completing this exercise, you will be able to:

- Create and use transformation parameters to define locations for input and output folder locations.
- Create and configure a CSV file input step.
- Create and configure a Switch / Case step that will send data to specific steps depending on the data contained in the incoming data stream.
- Create and configure a Text file output step that will create a new text file
- Create hops connected from a Switch / Case step that are configured to specific case values.

Continued on next page

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case, Continued

Create the Transformation

Step	Action
1	To create the new transformation, in the Spoon menubar, click **File | New | Transformation**.
2	To open the Transformation properties dialog, in the Canvas, double-click on an empty area.
3	Set the transformation properties for the **Transformation tab** according to the table below: **Property Name** / **Value** Transformation name: CsvInputTextOutput Description: Add a description of your choice. Directory: /public/PDI_Trn_Objects 📖 NOTE: This is in the repository.
4	Create two new parameters using the **Parameters tab** according to the table below: **Parameter Name** / **Default Value** KTR_DIR_INPUT: C:\pentahotraining\DataFiles\Input KTR_DIR_OUTPUT: C:\pentahotraining\DataFiles\Output 👍 TIP: Provide an optional description for each parameter to help others easily understand what they are used for.
5	To close the 'Transformation properties' dialog, click **[OK]**.

Continued on next page

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case, Continued

Create the Transformation, continued

Step	Action
6	To save the transformation: ■ Press **CTRL-S**. ■ At the 'Transformation properties' dialog, click **[OK]**. ■ At the 'Enter a comment' dialog, enter an optional comment, and then click **[OK]**. 🕮 NOTE: Now as you work on your transformation, you can easily save your work.

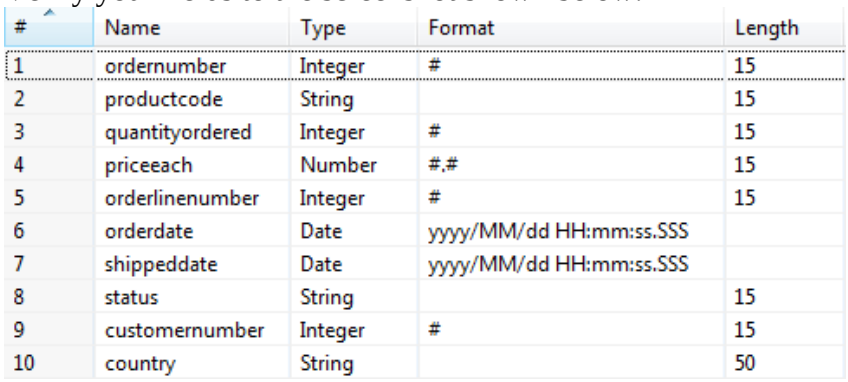
Create & Configure the CSV File Input Step

Step	Action
1	To create the first step, from the Input category of the Design tab, drag the CSV file input step onto the Canvas.
2	To open the step's properties dialog, double-click the step.
3	To set the step's properties, set them according to the table shown below: **Property Name** / **Value** Step Name: Read customer data from CSV Filename: ${KTR_DIR_INPUT}\Order_File.csv Delimiter: ; Lazy conversion: (unchecked)
4	To configure the fields for this step: Click the **[Get Fields]** button. At the 'Sample size' dialog, click **[OK]**. At the 'Scan results' dialog, click **[Close]**.

Continued on next page

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case, Continued

Create & Configure the CSV File Input Step, continued

Step	Action
5	Verify your fields to the screenshot shown below:  IMPORTANT: The CSV file input step must always be configured to read all of the fields in the file. You cannot choose a subset of the fields in the file and have it successfully read the data.
6	To close the step's properties dialog, click **[OK].**
7	Save the transformation.

#	Name	Type	Format	Length
1	ordernumber	Integer	#	15
2	productcode	String		15
3	quantityordered	Integer	#	15
4	priceeach	Number	#,#	15
5	orderlinenumber	Integer	#	15
6	orderdate	Date	yyyy/MM/dd HH:mm:ss.SSS	
7	shippeddate	Date	yyyy/MM/dd HH:mm:ss.SSS	
8	status	String		15
9	customernumber	Integer	#	15
10	country	String		50

Create & Configure the Switch/Case Step

Step	Action
1	To create the second step, from the Flow category of the Design tab, drag the Switch/Case step onto the Canvas.
2	To create a hop between the steps: Click on the first step to select it. Click and hold on the first step using the **middle mouse button**. Drag and release the hop onto the second step. Choose Main output of step
3	To open the step's properties dialog, double-click the new step.

Continued on next page

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case, Continued

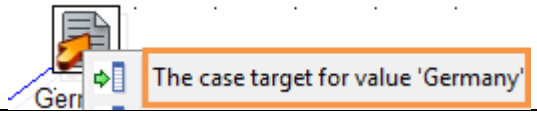
Create & Configure the Switch/Case Step, continued

Step	Action
4	To set the step's properties, set them according to the table shown below: **Property Name** / **Value** Step Name: Switch / Case on country Field name to switch: country Use string contains comparison: (checked) Case value data type: String
5	Set the Case values grid, according to the table below: **#** / **Value** 1: Germany 2: France 3: Australia 4: USA 🕮 NOTE: The remaining properties of this step will automatically get set as your build and configure the rest of your transformation.
6	To close the step's properties dialog, click **[OK]**.
7	Save the transformation.

Continued on next page

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case, Continued

Create & Configure the Germany Text File Output Step

Step	Action
1	To create the Germany Text file output step, from the Output category of the Design tab, drag the Text file output step onto the Canvas above and to the right of the Switch / Case step.
2	Create a hop between the Switch/Case and Text file output steps.
3	Set the Text file output step's properties according to the table shown below: **Property Name** / **Value** Step Name: Germany Filename: ${KTR_DIR_OUTPUT}\Country_Germany
4	Create a hop between the Switch/Case and Germany steps, and when prompted, choose The case target for value 'Germany'. 
5	To configure the field values for this step: Open the Germany step's properties dialog. Click on the Fields tab. Click [Get Fields]. Click **[OK]**.
6	Save the transformation.

Create & Configure the Remaining Output Steps

Step	Action
1	Create three additional Text file output steps given the step names below and configure their properties Fields tabs as you did for the Germany step. France Australia USA

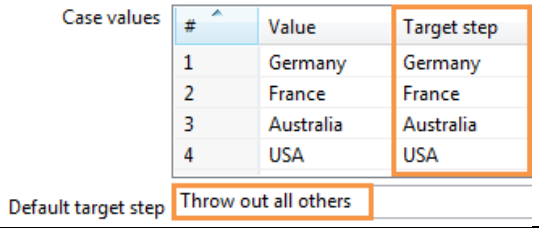
Continued on next page

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case, Continued

Create & Configure the Remaining Output Steps, continued

Step	Action
2	Create three new hops according to the table below: Source Step / Destination Step / Case Target Value Switch / Case on country / France / France Switch / Case on country / Australia / Australia Switch / Case on country / USA / USA
3	Save the transformation.

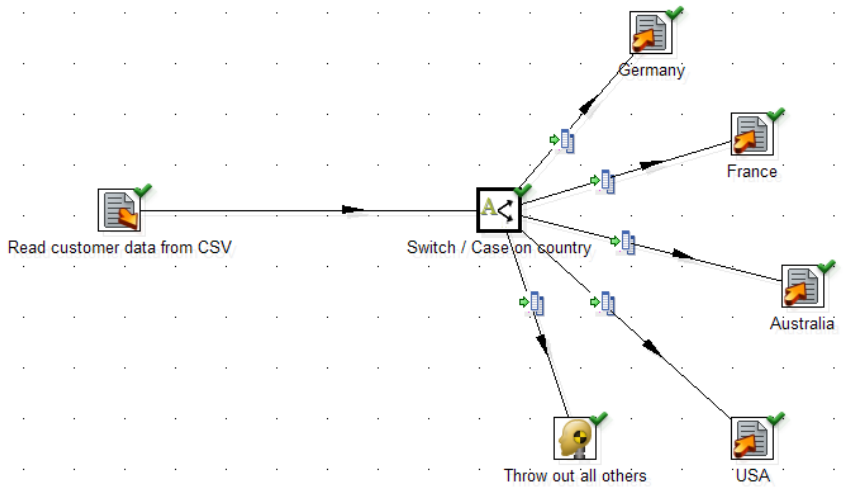
Create & Configure the Dummy Step

Step	Action
1	Create a Dummy (do nothing) step and configure it according to the table below: Property Name / Value Step name / Throw out all others
2	Create a new hop according to the table below: Source Step / Destination Step / Case Target Value Switch / Case on country / Throw out all others / The default target step
3	To see how the Switch/Case step's properties have been updated as you configured the last steps and hops, open the Switch/Case step's properties dialog and compare with the screenshot below: 

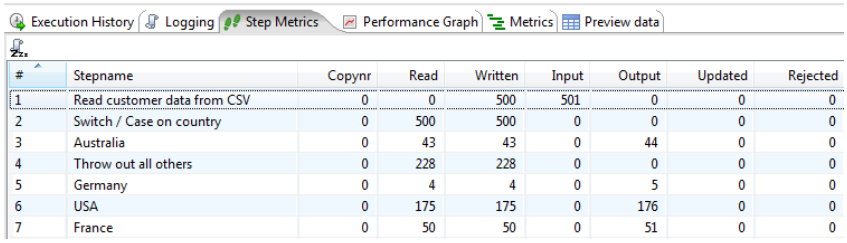
Continued on next page

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case, Continued

Create & Configure the Dummy Step, continued

Step	Action
4	The transformation should look similar to the screenshot below: 
5	Save the transformation.

Execute the Transformation

Step	Action
1	To run the transformation, press **F9**, and then click **[Launch]**.
2	There should be no errors and the **Step Metrics** tab should look similar to the screenshot shown below: 

Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data

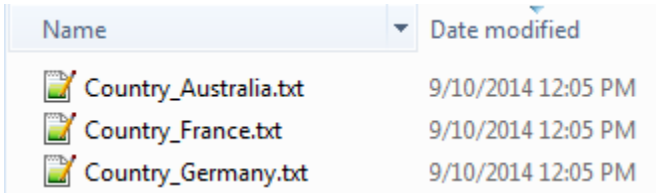
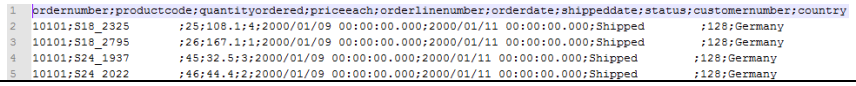
#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected
1	Read customer data from CSV	0	0	500	501	0	0	0
2	Switch / Case on country	0	500	500	0	0	0	0
3	Australia	0	43	43	0	44	0	0
4	Throw out all others	0	228	228	0	0	0	0
5	Germany	0	4	4	0	5	0	0
6	USA	0	175	175	0	176	0	0
7	France	0	50	50	0	51	0	0

🕮 NOTE: Notice the Output field contains data. This is indicates the number of lines output to the text files that it created.

Continued on next page

Exercise 2 – CSV Input to Multiple Text Output Using Switch/Case, Continued

Execute the Transformation, continued

Step	Action
3	To verify the text files were created, navigate to the folder shown below and verify the creation of the files as show in in the example screenshot: C:\pentahotraining\DataFiles\Output 
4	Open each one of the text files and notice how the data corresponding to each country is written to the appropriate country's text file. Example of Country_Germany.txt 

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Exercises`

Completed transformation:
`EX2_CsvInputTextOutput.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Exercise 2 Advanced – CSV Input to Multiple Text Output Using Switch/Case

Introduction

This is an advanced version of Exercise 2. It is the same exercise, but without the detailed guidance. You may choose to do this advanced exercise rather than the standard version of Exercise 2.

Instructions

Create a transformation that can do the following:

- Read the Order_File.csv file located at C:\pentahotraining\DataFiles\Input
- Separate the order details by country.
- Output the order details to four text files by country as follows:
  - Germany
  - France
  - Australia
  - USA
- All other countries order data should be discarded.
- The text files should be written to the following location: C:\pentahotraining\DataFiles\Output
- Use transformation parameters to define the input and output file locations.

Steps Used

Use only the steps shown below to complete this exercise:

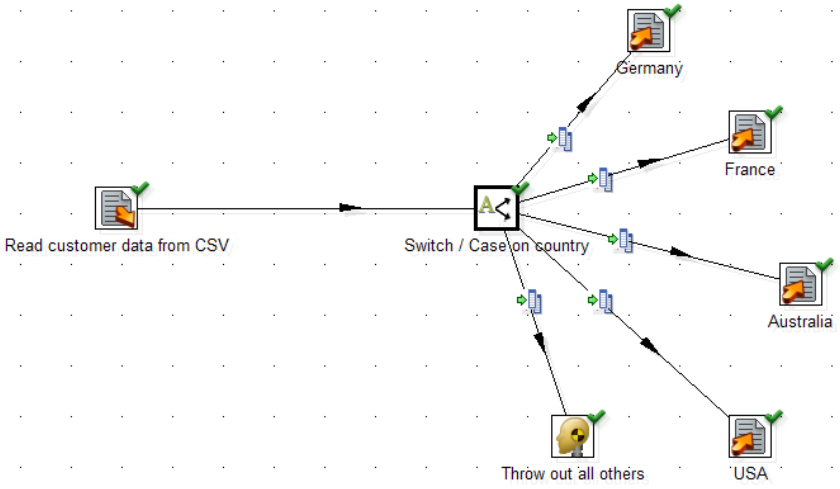
- CSV file input
- Switch / Case
- Text file output
- Dummy (do nothing)

Continued on next page

Exercise 2 Advanced – CSV Input to Multiple Text Output Using Switch/Case, Continued

Finished Transformation

The completed transformation should resemble the screenshot shown below:



Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

End of Exercise

Congratulations! You have completed this exercise.

Exercise 3 – Serializing Multiple Text Files

Introduction

In this exercise, you will create a transformation that reads the multiple text files you created in the previous exercise. Then, the data is sent to a serialize step that creates a binary (serialized) file. A regular expression is used, along with a parameter, to determine the correct files to read.

📖 NOTE

For those looking for more of a challenge, try the advanced version of this exercise. It is the same exercise, but without the detailed guidance. You will find it in this workbook immediately following this exercise.

Objectives

After completing this exercise, you will be able to:

- Read multiple text files from a parameterized directory using the Text file input step.
- Use a regular expression to specify a specific pattern of filename.
- Create and configure a Serialize step to create binary file from an input stream.

Prerequisites

The multiple text files created from executing the previous exercise.

Continued on next page

Exercise 3 – Serializing Multiple Text Files, Continued

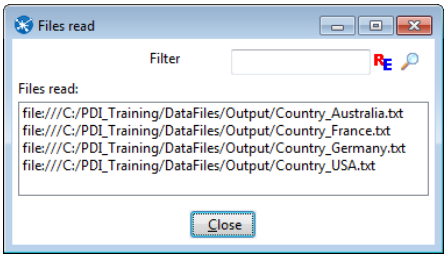
Create the Transformation

Step	Action
1	To create the new transformation, in the Spoon menubar, click **File | New | Transformation**.
2	To open the 'Transformation properties' dialog, in the Canvas, double-click on an empty area.
3	Set the transformation properties for the **Transformation tab** according to the table below: Property Name: Value Transformation name: SerializeMultipleTextFiles Description: Add a description of your choice. Directory: /public/PDI_Trn_Objects 📖 NOTE: This is in the repository.
4	To close the 'Transformation properties' dialog, click **[OK]**.
5	To save the transformation: ■ Press **CTRL-S**. ■ At the 'Transformation properties' dialog, click **[OK]**. ■ At the 'Enter a comment' dialog, enter an optional comment, and then click **[OK]**. 📖 NOTE: Now as you work on your transformation, you can easily save your work.

Continued on next page

Exercise 3 – Serializing Multiple Text Files, Continued

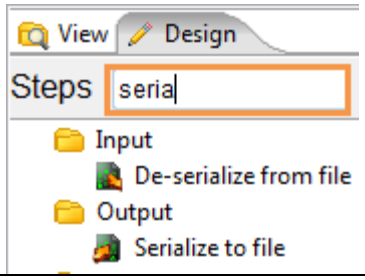
Create & Configure the Text File Input Step

Step	Action
1	To create the first step, from the Input category of the Design tab, drag the Text file input step onto the Canvas.
2	To open the step's properties dialog, double-click the step.
3	To set the step's properties, set them according to the table shown below: Property Name: Value Step Name: Read multiple text files Selected files: File/Directory: ${DIR_OUTPUT}\ Selected files: Wildcard (RegExp): Country_.*\.txt
4	To check that the source directory and filename regular expression are correct, and that the files exist, click the **[Show filename(s)…]** button. You should see the following dialog and content:  Click **[Close]**.
5	To configure the fields for this step: Click the **Fields** tab. Click the **[Get Fields]** button. At the 'Sample size' dialog, click **[OK]**. At the 'Scan results' dialog, click **[Close]**.
6	To preview the data, click the [Preview] button and verify the data returned contains all of the countries you used as input in the 'country' column.
7	To close the step's properties dialog, click **[OK]**.
8	Save the transformation.

Continued on next page

Exercise 3 – Serializing Multiple Text Files, Continued

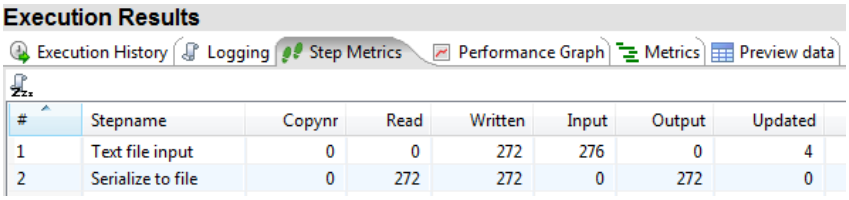
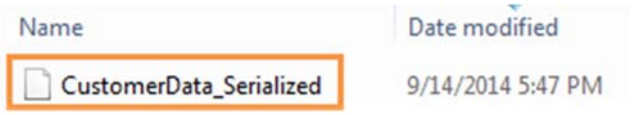
Create & Configure the Serialize Step

Step	Action
1	To use the search box and find the next step to add to your transformation, from the Design tab, enter the following text into the search box: “seria” 
2	To add the step, from the Output category of the Design tab, drag the Serialize to file step onto the Canvas.
3	To open the step’s properties dialog, double-click the step.
4	To set the step’s properties, set them according to the table shown below: Property Name: Value Filename: ${DIR_OUTPUT}\CustomerData_Serialized
5	Create a new hop according to the table below: Source Step: Read multiple text files Destination Step: Serialize to file
6	Save the transformation.

Continued on next page

Exercise 3 – Serializing Multiple Text Files, Continued

Execute the Transformation

Step	Action
1	Run the transformation.
2	There should be no errors and the Step Metrics tab should look similar to the screenshot shown below:  \# | Stepname | Copynr | Read | Written | Input | Output | Updated 1 | Text file input | 0 | 0 | 272 | 276 | 0 | 4 2 | Serialize to file | 0 | 272 | 272 | 0 | 272 | 0
3	To verify the serialized file was created, navigate to the folder shown below and verify the creation of the file as show in in the example screenshot: C:\pentahotraining\DataFiles\Output 

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Exercises`

Completed transformation:

`EX3_SerializeMultipleTextFiles.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Exercise 3 Advanced – Serializing Multiple Text Files

Introduction

This is an advanced version of Exercise 3. It is the same exercise, but without the detailed guidance. You may choose to do this advanced exercise rather than the standard version of Exercise 3.

Instructions

Create a transformation that has the following functionality:

- Read all files located in your parameterized output folder that match the following filename pattern: Begin with 'Country_' and have the 'txt' extension.
- Use a regular expression for reading the correct file names.
- Serialize all the files and output to your parameterize output folder.
- The output filename is: CustomerData_Serialized

Steps Used

Use only the steps shown below to complete this exercise:

- Text file input
- Serialize to file

Finished Transformation

The completed transformation should resemble the screenshot shown below:

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

End of Exercise

Congratulations! You have completed this exercise.

Exercise 4 – De-serializing a File

Introduction

In this exercise, you will create a transformation that reads serialized file you created in the previous exercise and de-serializes. Then, the data is sent to a Text file output step.

📖 NOTE

For those looking for more of a challenge, try the advanced version of this exercise. It is the same exercise, but without the detailed guidance. You will find it in this workbook immediately following this exercise.

Objectives

After completing this exercise, you will be able to:

- De-serialize a file.
- Create a text file output with minimal width delimited data and customized fields.

Prerequisites

The serialized file created from executing the previous exercise.

Continued on next page

Exercise 4 – De-serializing a File, Continued

Create the Transformation

Step	Action
1	To create the new transformation, in the Spoon menubar, click **File** | **New** | **Transformation**.
2	To open the Transformation properties dialog, in the Canvas, **double-click** on an empty area.
3	Set the transformation properties for the **Transformation tab** according to the table below:

Property Name	Value
Transformation name	DeserializeFile
Description	Add a description of your choice.
Directory	/public/PDI_Trn_Objects 🕮 NOTE: This is in the repository.

Step	Action
4	To close the 'Transformation properties' dialog, click **[OK]**.
5	To save the transformation: ■ Press **CTRL-S**. ■ At the 'Transformation properties' dialog, click **[OK]**. ■ At the 'Enter a comment' dialog, enter an optional comment, and then click **[OK]**.

Continued on next page

Exercise 4 – De-serializing a File, Continued

Create & Configure the De-serialize Step

Step	Action
1	To create the first step, from the **Input** category of the **Design tab**, drag the **De-serialize a file** step onto the Canvas.
2	To open the step's properties dialog, double-click the step.
3	To set the step's properties, set them according to the table shown below: **Property Name** / **Value** Step Name: De-serialize Country Order Data Filename: C:\pentahotraining\DataFiles\Output\CustomerData_Serialized
4	To close the step's properties dialog, click **[OK]**.
5	Save the transformation.

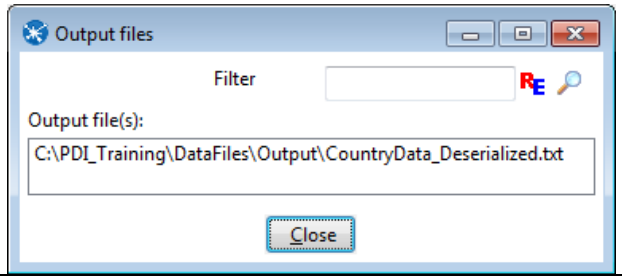
Create & Configure the Text file output Step

Step	Action
1	To create the Text file output step, from the **Output** category of the **Design** tab, drag the **Text file output** step onto the Canvas above and to the right of the Switch / Case step.
2	Create a hop between the De-serialize and Text file output steps.
3	Set the Text file output step's properties according to the table shown below: **Property Name** / **Value** Step Name: Combined Country Data to Text Filename: C:\pentahotraining\DataFiles\Output\CountryData_Deserialized

Continued on next page

Exercise 4 – De-serializing a File, Continued

Create & Configure the Text file output Step, continued

Step	Action
4	To verify the correct filename will be created when this transformation is run, click **[Show filename(s)…]**. 
5	To add all of the fields contained in the input file to this step: ■ Click on the **Fields** tab. ■ Click [Get Fields].
6	The text file you want to output will only contain a subset of these fields, because all are not needed. To configure the fields tab for only the fields needed, delete all but the following fields: ordernumber status country
7	To configure the text file output to have no trailing spaces in each column's data, click **[Minimal width]**.
8	To close the 'Text file output' dialog, click **[OK]**.
9	Save the transformation.

Continued on next page

Exercise 4 – De-serializing a File, Continued

Preview the De-serialized data in the stream

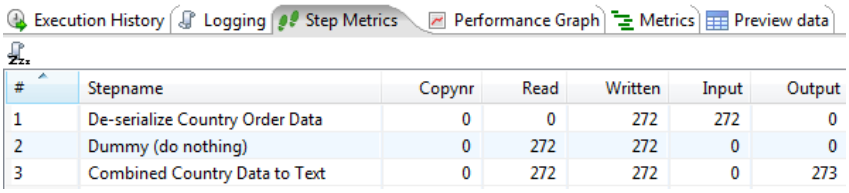
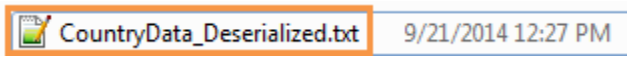
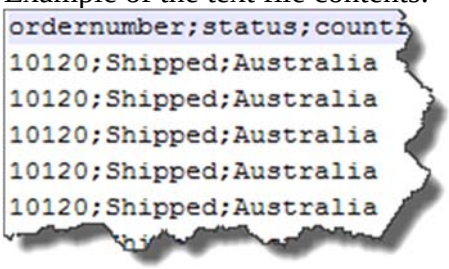
The transformation is nearly complete, but you may wish to preview data that is coming from the De-serialize step before actually writing a text output file.

Step	Action
4	To preview the data coming from the De-serialize step: ■ **Select** the "Text file output" step. ■ Click the **Preview** button in the Canvas toolbar. ■ Click **[Quick Launch]** and preview the data. ■ Click **[Close]** to close the 'Examine preview' data dialog. ■ Click **[Close]** to close the 'Select the preview' step dialog.

Continued on next page

Exercise 4 – De-serializing a File, Continued

Execute the Transformation

Step	Action
1	Run the transformation.
2	There should be no errors and the 'Step Metrics' tab should look similar to the screenshot shown below:  Execution History | Logging | Step Metrics | Performance Graph | Metrics | Preview data \# | Stepname | Copynr | Read | Written | Input | Output 1 | De-serialize Country Order Data | 0 | 0 | 272 | 272 | 0 2 | Dummy (do nothing) | 0 | 272 | 272 | 0 | 0 3 | Combined Country Data to Text | 0 | 272 | 272 | 0 | 273 TIP: The value in the 'Output' column shows that in fact data was output to a file or table. In this case 273 lines, were written to a text file.
3	To verify the text file was created, navigate to the folder shown below and verify the creation of the file as show in in the example screenshot: C:\pentahotraining\DataFiles\Output  CountryData_Deserialized.txt 9/21/2014 12:27 PM
4	To verify the text file contains the de-serialized (plain text) data, double-click the file. Example of the text file contents:  ordernumber;status;countr 10120;Shipped;Australia 10120;Shipped;Australia 10120;Shipped;Australia 10120;Shipped;Australia 10120;Shipped;Australia

Continued on next page

Exercise 4 – De-serializing a File, Continued

Solution Details

The solution to this exercise can be obtained using the details below:

Location: `C:\pentahotraining\Solutions\DeserializeFile\`

Completed transformation:

`EX4_DeserializeFile.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Exercise 4 Advanced – De-Serializing Multiple Text Files

Introduction

This is an advanced version of Exercise 4. It is the same exercise, but without the detailed guidance. You may choose to do this advanced exercise rather than the standard version of Exercise 4.

Instructions

Create a transformation that has the following functionality:

- De-serialize the file output in Exercise 3, 'CustomerData_Serialized'.
- Use a Dummy (do nothing) step to preview the file and verify it was properly de-serialized.
- Output only the fields shown below to a text file in your output directory.
  - ordernumber
  - status
  - country

Steps Used

Use only the steps shown below to complete this exercise:

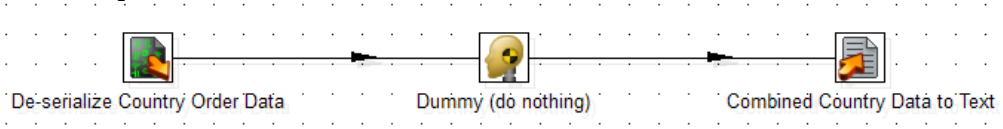
- De-serialize from file
- Dummy (do nothing)
- Text file output

📖 NOTE

The "Dummy (do nothing)" step is optional. It is not necessary for the transformation to work correctly.

Finished Transformation

The completed transformation should resemble the screenshot shown below:



Prerequisites

The serialized file created from executing the previous exercise.

End of Exercise

Congratulations! You have completed this exercise.

Guided Demo 7 – Connections & the Database Explorer

Introduction

In this guided demonstration, you will prepare your environment to use a MySQL database copying the MySQL driver to the proper location and starting the MySQL database. Then you will create a database connection and use the Database Explorer.

Objectives

In this guided demonstration, you will:

- Understand the placement of database drivers and how to start the MySQL database server.
- Create a database connection
- Use Database Explorer to interact with a data source

Prerequisites

You must complete the prior exercise on installing and configuring Pentaho Data Integration in order to complete this exercise. You must also have access to course files required (if any).

Part I: Adding the MySQL Driver to the PDI Environment

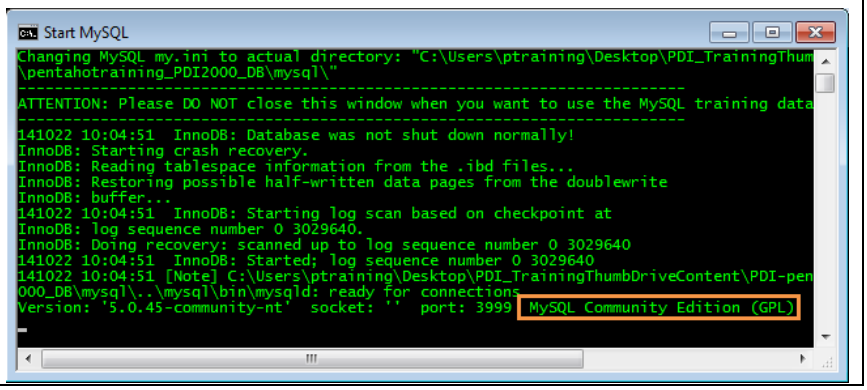
You will be using a MySQL database. Pentaho Data Integration does not ship with the MySQL database driver. In order for your environment to function correctly, this driver needs to be added to the PDI directory structure. Therefore, before this guided demo can really begin, you need to make sure the MySQL driver is in the proper location, and that the database server is started.

Step	Action
1	Save any of your current work and close Spoon.
2	Locate the mysql-connector-java-5.1.17-bin.jar file in the following folder: **C:\pentahotraining\Software\MySQL Driver**

Continued on next page

Guided Demo 7 – Connections & the Database Explorer, Continued

Part I: Adding the MySQL Driver to the PDI Environment, continued

Step	Action
3	Copy the mysql-connector-java-5.1.17-bin.jar file into the following *two* folders: **C:\Pentaho\design-tools\data-integration\lib** **C:\Pentaho\server\data-integration-server\tomcat\lib**
4	Run the Start MySQL shortcut located in the following folder: **C:\pentahotraining\Software\MySQL Driver** A console will open and the MySQL database server will start. If you are presented with a firewall screen, click **[Allow]**. 

CAUTION Do not close this window, otherwise MySQL will stop and you cannot connect to the MySQL database.

Part II: Database Connections In part II of this exercise, you create a database connection in PDI.

Continued on next page

Guided Demo 7 – Connections & the Database Explorer,
Continued

Creating a Database Connection

To create a database connection in PDI:

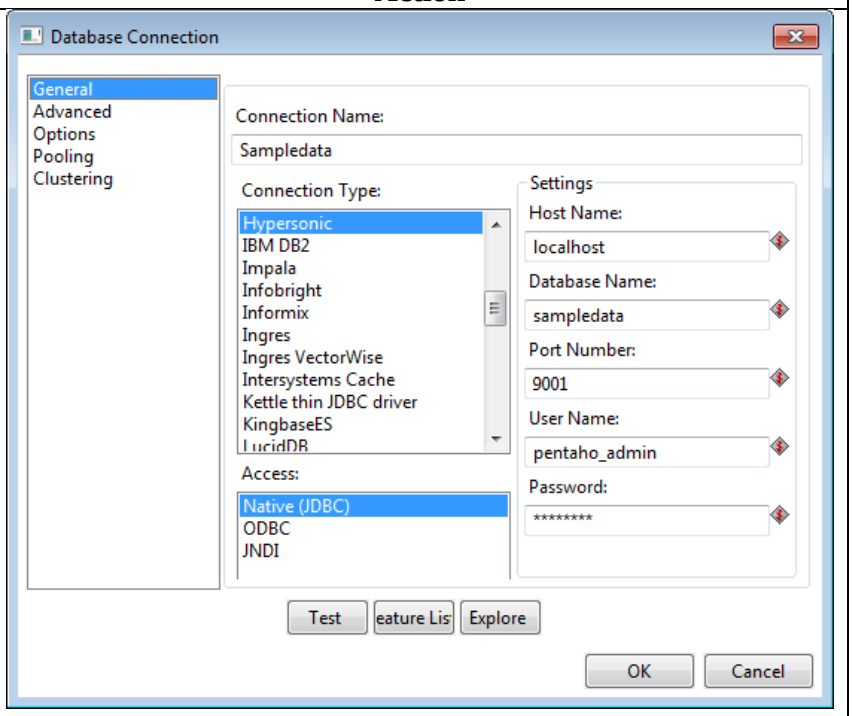
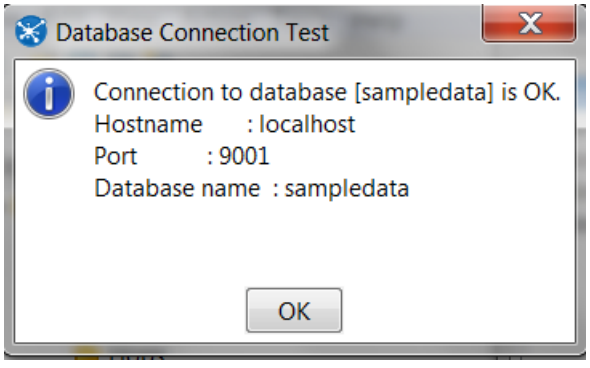
Step	Action
1	Open an existing transformation
2	Save the transformation as **ConnectExplore**
3	In Pentaho Data Integration, switch from the Design tab to the **View** tab. 
4	Right click on Database connection and select new. Fill in the dialogue box as follows.
5	In the 'Database Connection' dialog, type or select the following:

Field	Value
Connection Name	sampledata
Connection Type	Hypersonic
Access	Native (JDBC)
Host Name	localhost
Database Name	sampledata
Port Number	9001
User Name	pentaho_admin
Password	password

Continued on next page

Guided Demo 7 – Connections & the Database Explorer, Continued

Creating a Database Connection (continued)

Step	Action
7 (cont)	
8	Click **Test**. A pop-up dialog shows the test result: 
9	Click **[OK]** to close the "Database Connection Test" dialog.

Continued on next page

Guided Demo 7 – Connections & the Database Explorer,
Continued

Creating a Database Connection (continued)

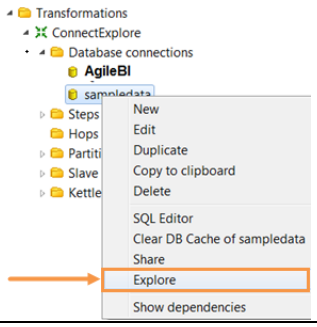
Step	Action
10	In the 'Database Connection' dialog, click **Feature List**.
11	Notice the connection properties. In particular, notice the **Driver class** and **URL** values. 19 Driver class org.hsqldb.jdbcDriver 20 URL jdbc:hsqldb:hsql://localhost:9001/sampledata
12	Click on the Parameter column header, that will sort the display by Parameters.
13	Click **[OK]** to close the "Database Connection" dialog.
14	On the **View** tab, expand **Database Connections**, right-click **sampledata.**
15	Save the transformation.

Part III: Database Explorer

In part III of this guided demo, you will use the Database Explorer tool to interact with a configured data source.

Using Database Explorer

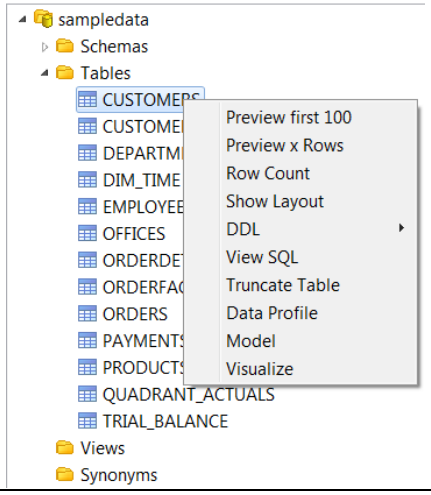
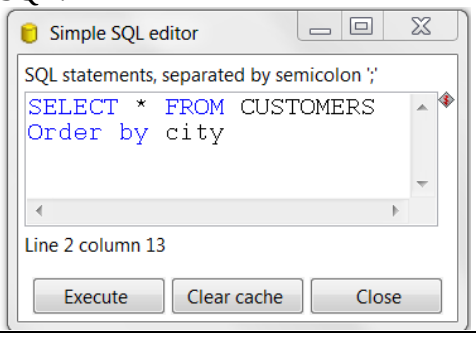
To use Database Explorer:

Step	Action
1	If necessary, on the **View** tab, expand **Database Connections**.
2	Right-click **sampledata** and choose **Explore** from the menu options. 

Continued on next page

Guided Demo 7 – Connections & the Database Explorer, Continued

Using Database Explorer, continued

Step	Action
3	In the **Database Explorer** window, expand **pentaho_oltp.**
4	Right-click the **customers** table and choose **Preview first 100** from the menu options. 
5	Examine the customer data.
6	Select '**View SQL'**. The simple SQL editor appears. Type this SQL: 
7	Click **[Execute]**.
8	Close the Simple SQL editor.
9	Click **[OK]** to close Database Explorer. (You may need to expand the size of the dialog to view the **[OK]** button).

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Exercise 5 – Reading & Writing to Database Tables

Introduction

This exercise is designed to introduce you to various methods of interacting with databases using Pentaho Data Integration. First, you create a database connection, explore a data source, and create transformations that use various data input and output steps including: Table input, Table output, Text file output, and CSV file input.

📖 NOTE

For those looking for more of a challenge, try the advanced version of this exercise. It is the same exercise, but without the detailed guidance. You will find it in this workbook immediately following this exercise.

Objectives

After completing this exercise, you will be able to:

- Create a database connection.
- Use Database Explorer to interact with a data source.
- Create a transformation that uses the Table input and Table output steps.
- Create a transformation that uses the Text file output step.

Prerequisites

You must complete the prior exercise on installing and configuring Pentaho Data Integration in order to complete this exercise. You must also have access to course files required (if any).

Continued on next page

Exercise 5 – Reading & Writing to Database Tables, Continued

Part I: Database Connections

In part I of this exercise, you create a database connection in PDI.

Creating a Database Connection

To create a database connection in PDI:

Step	Action
1	Start Spoon.
2	Choose **File | New | Transformation** to create a new transformation. (You can also click the **New File** icon and choose **Transformation**).
3	Save the transformation in the repository using File name: 'TableInputOutput'.
4	In Pentaho Data Integration, switch from the Design tab to the **View** tab. 
5	Expand **Transformations > TableInputOutput**, right-click **Database connections** and choose **New**.

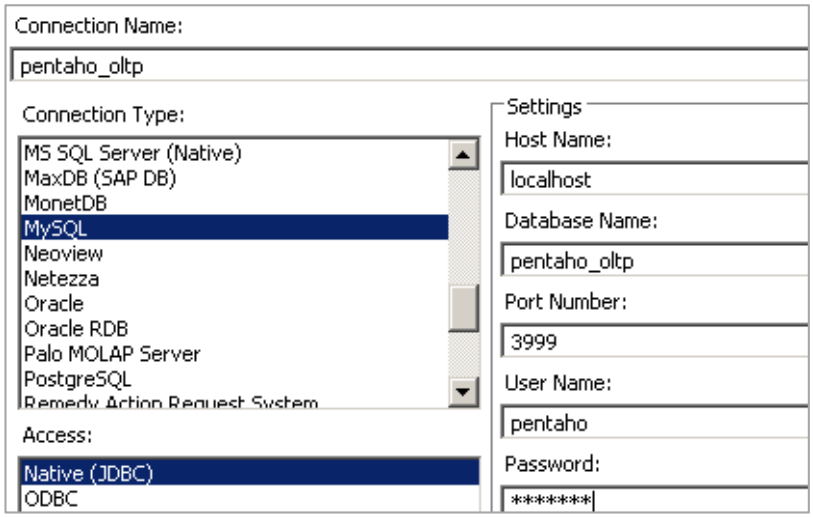
Continued on next page

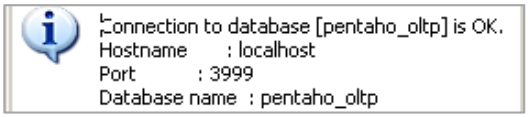
Exercise 5 – Reading & Writing to Database Tables, Continued

Creating a Database Connection (continued)

Step	Action
6	In the 'Database Connection' dialog, type or select the following:

Field	Value
Connection Name	pentaho_oltp
Connection Type	MySQL
Access	Native (JDBC)
Host Name	localhost
Database Name	pentaho_oltp
Port Number	3999
User Name	pentaho
Password	pentaho



7	Click **Test**. A pop-up dialog shows the test result: 
8	Click **[OK]** to close the "Database Connection Test" dialog and click **[OK]** to close the "Database Connection" dialog.

Continued on next page

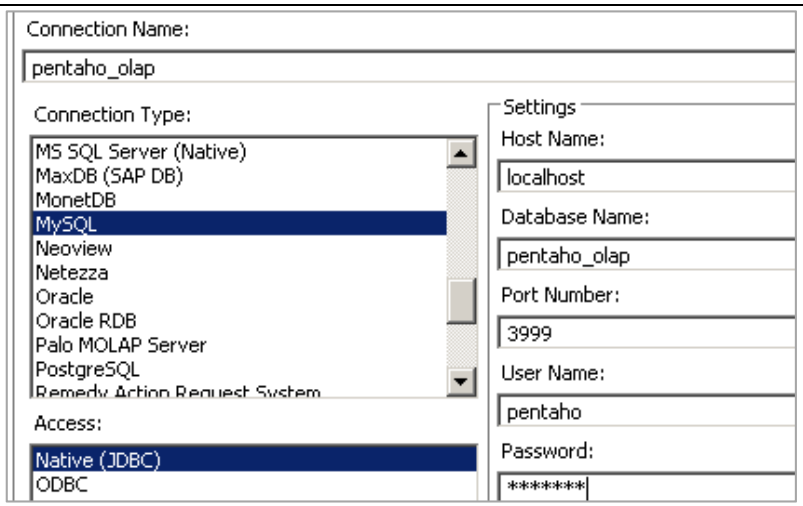
Exercise 5 – Reading & Writing to Database Tables, Continued

Creating a Database Connection (continued)

Step	Action
9	On the **View** tab, expand **Database Connections**, right-click **pentaho_oltp** and choose **Share** from the context menu. When a connection is shared, it appears in bold text.
10	Repeat the previous steps to create a connection to the **pentaho_olap** database using the following values:

Field	Value
Connection Name	pentaho_olap
Connection Type	MySQL
Access	Native (JDBC)
Host Name	localhost
Database Name	pentaho_olap
Port Number	3999
User Name	pentaho
Password	pentaho

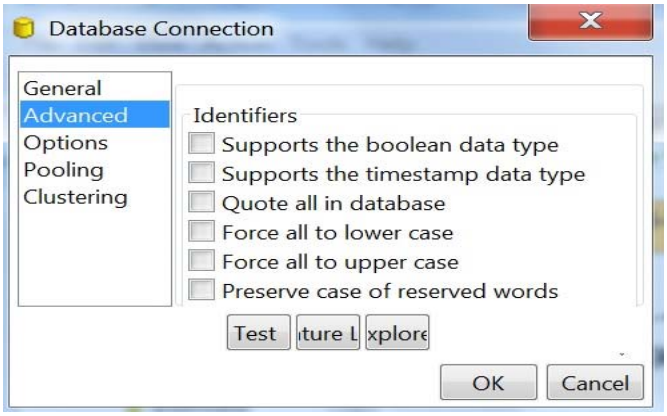
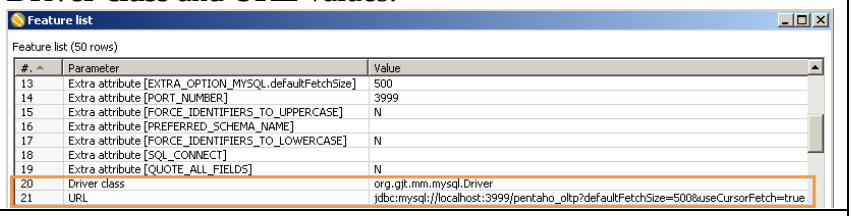
11



Continued on next page

Exercise 5 – Reading & Writing to Database Tables, Continued

Creating a Database Connection (continued)

Step	Action
12	Click on the Advanced tab on the left side of the dialog, and then unclick the first two check boxes. 
13	Click **Test** to verify the connection.
14	Click **[OK]** to close the "Database Connection Test" dialog.
15	Double-click the **pentaho_olap** connection to open it.
16	In the 'Database Connection' dialog, click **Feature List**.
17	Notice the connection properties. In particular, notice the **Driver class** and **URL** values. 
18	Click **[OK]** to close the "Database Connection" dialog.
19	On the **View** tab, expand **Database Connections**, right-click **pentaho_olap** and choose **Share** from the context menu. Notice that when the connection is shared, it appears in bold text.
20	Save the transformation.

Continued on next page

Exercise 5 – Reading & Writing to Database Tables, Continued

Part II: Database Explorer

In part II of this exercise, you use the Database Explorer tool to interact with a configured data source. To use Database Explorer:

Using Database Explorer

Step	Action
1	If necessary, on the **View** tab, expand **Database Connections**.
2	Right-click **pentaho_oltp** and choose **Explore** from the menu options.
3	In the **Database Explorer** window, expand **pentaho_oltp** > **Tables**.
4	Right-click the **customers** table and choose **Preview first 100** from the menu options.
5	Examine the customer data and click **Close** when you are finished.
6	As time permits, explore the data in the other tables in pentaho_oltp.
7	Click **[OK]** to close Database Explorer. (You may need to expand the size of the dialog to view the **[OK]** button).

Part III: Using Table Input & Output

In part III of this exercise, you create a transformation that uses the Table input and Table output steps. The transformation is used to transfer data between databases – a common procedure in data warehousing.

Using Table Input & Input Steps

To create a transformation using Table input and output steps:

Step	Action
1	Switch to the **Design** tab.
2	Drag an **Input** > **Table input** step onto the canvas.
3	Drag an **Output** > **Table output** step onto the canvas.
4	Create a **hop** between **Table input** and **Table output**.
5	Double-click **Table input**.
6	In the 'Table input' dialog, for **Connection**, choose **pentaho_oltp** from the drop-down list.
7	Click the **Get SQL select statement** button.
8	In the 'Database Explorer' dialog, expand **pentaho_oltp** > **Tables**, select **orderdetails** and click **[OK]**.

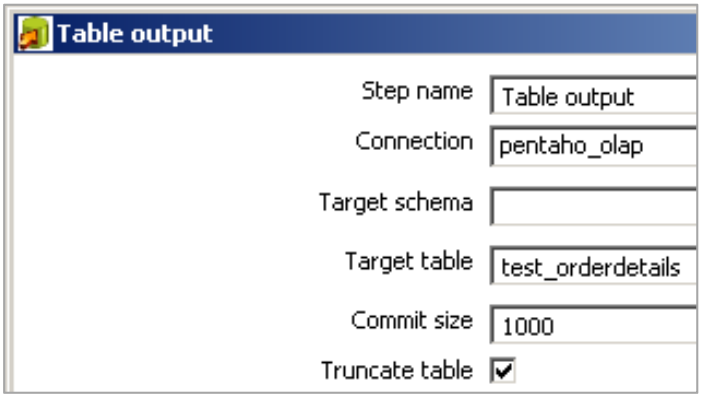
Continued on next page

Exercise 5 – Reading & Writing to Database Tables, Continued

Using Table Input & Input Steps (continued)

Step	Action

9	When prompted to include field names in the select statement, click **Yes**. The resulting SQL statement should look like the following:
	`SELECT`
	`  ordernumber`
	`, productcode`
	`, quantityordered`
	`, priceeach`
	`, orderlinenumber`
	`FROM orderdetails`

10	Click **[OK]** to close the Table input dialog.
11	Double-click the **Table output** step.
12	In the **Table output** dialog, type or choose the following: **Field** / **Value** Connection: `pentaho_olap` Target schema: `[leave blank]` Target table: `test_orderdetails` Commit size: `1000` Truncate table: `[checked]` 
13	At the bottom of the 'Table output' dialog, click the **SQL** button.

Continued on next page

Exercise 5 – Reading & Writing to Database Tables, Continued

Using Table Input & Input Steps (continued)

Step	Action

14	Verify the syntax of the SQL statement looks like the following:
	`CREATE TABLE test_orderdetails`
	`(`
	`  ordernumber INT`
	`, productcode VARCHAR(15)`
	`, quantityordered INT`
	`, priceeach DOUBLE`
	`, orderlinenumber INT`
	`)`
	`;`

15	In the 'Simple SQL editor' dialog, click **Execute** to create the table from the SQL statement.
16	When you receive the **Results** dialog, verify the SQL statements were executed and click **[OK]**.
17	When you are returned to the **Simple SQL editor**, click **Close**.
18	When you are returned to the 'Table ouput' dialog, click **[OK]**.
19	Save your work.
20	Click the **Run this transformation** icon, then click **Launch**.
21	Switch to the **Step Metrics** view.
22	The Table input step should write 23640 rows, while the Table output step should both read and write 23640 rows. Execution History | Logging | Step Metrics | Performance Graph #. | Stepname | Copynr | Read | Written | Input | Output 1 | Table input | 0 | 0 | 23640 | 23640 | 0 2 | Table output | 0 | 23640 | 23640 | 0 | 23640
23	Double-click the **Table input** step.

24	In the SQL statement, type a line below **, priceeach** as follows:
	`,  quantityordered * priceeach as total`
	`SELECT`
	`  ordernumber`
	`, productcode`
	`, quantityordered`
	`, priceeach`
	`, quantityordered * priceeach as total`
	`, orderlinenumber`
	`FROM orderdetails`

25	Click **[OK]**.

Continued on next page

Exercise 5 – Reading & Writing to Database Tables, Continued

Using Table Input & Input Steps (continued)

Step	Action
26	Because you already executed the SQL code to create the table, running the transformation will produce errors. Double-click the **Table output** step to alter the database.
27	In the 'Table output' dialog, click the **SQL** button.
28	The Simple SQL editor should display the ALTER TABLE code necessary to change the database table: `ALTER TABLE test_orderdetails ADD total DOUBLE;`
29	In the **Simple SQL editor**, click **Execute**.
30	When you receive the 'Results' dialog indicating the SQL statement was executed, click **[OK]** to close it.
31	Click **Close** to close the Simple SQL editor.
32	Click **[OK]** to close the 'Table output' dialog.
33	Save your work.
34	Click the **Run this transformation** icon, then click **Launch**.
35	Switch to the **Step Metrics** view and note the results. The results should be the same as the previous run (23640 rows read/written).
36	(Optional) As time permits, use Database Explorer to examine the contents of the **pentaho_olap > test_orderdetails** table.
37	Leave the transformation open.

Part IV: Text Output

In part IV of this exercise you create a transformation that uses the Text file output step to write data to a text file.

Using the Text File Output Step

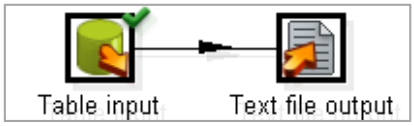
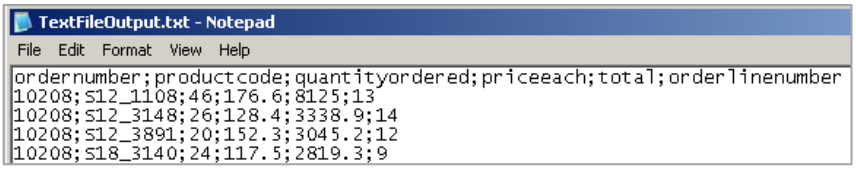
To create a transformation that writes data to the file system using a Text file output step:

Step	Action
1	Choose **File | Save As** from the menu options.
2	Save the file in the repository using **Transformation name**: **`TextFileOutput`**.
3	If you receive the 'Transformation Properties' dialog, confirm the **Transformation name** and click **[OK]**.

Continued on next page

Exercise 5 – Reading & Writing to Database Tables, Continued

Using the Text File Output Step (continued)

Step	Action
4	Delete the **Table output** step from the canvas and click **Yes** to confirm.
5	Drag an **Output** > **Text file output** step onto the canvas.
6	Create a **hop** between **Table input** and **Text file output**. 
7	Double-click the **Text file output** step.
8	In the **Filename** field, type (or browse to): **`C:\pentahotraining\DataFiles\Output\`** Use file name: **`TextFileOutput`** 🕮 NOTE: Leaving the "Fields" tab empty writes all fields to the file.
9	Click **[OK]** and save your work.
10	Click the **Run this transformation** icon, then click **Launch**.
11	When the transformation is complete, open and examine: 
12	Close "TextFileOutput.txt" and close the transformation.

Solution Details

The solution to this exercise can be obtained using the details below:

Location: `C:\pentahotraining\Solutions\Exercises`

Completed transformations:
`EX5_TableInputOutput.ktr`
`EX5_TextFileOutput.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Exercise 5 Advanced – Reading & Writing to Database Tables

Introduction

This is an advanced version of Exercise 5. It is the same exercise, but without the detailed guidance, and the addition of using a parameter. You may choose to do this advanced exercise rather than the standard version of Exercise 5.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Instructions & Objectives

This exercise contains several parts. Please read each part section below to see the requirements and instructions necessary for completing each part this advanced exercise.

After completing all parts of this exercise, you will be able to:

- Create a database connection.
- Use Database Explorer to interact with a data source.
- Create a transformation that uses the Table input and Table output steps. Choose a table name that is unique and not already exist.
- Create a transformation that uses the Text file output step.

Continued on next page

Exercise 5 Advanced – Reading & Writing to Database Tables, Continued

Part I

Create and test a database connection with the following connection properties:

Field	Value
Connection Name	pentaho_oltp
Connection Type	MySQL
Access	Native (JDBC)
Host Name	localhost
Database Name	pentaho_oltp
Port Number	3999
User Name	pentaho
Password	pentaho

Part II

Use the database connection created in Part I, and the Database Explorer to preview the first 100 rows of the 'customers' table.

Part III

Create a transformation that has the following functionality:

- Loads the contents of one table into another according to the exercise details below.

Exercise Detail	Value
Source Database	pentaho_oltp
Source Table	orderdetail
Destination Database	pentaho_otap
Destination Table	test_orderdetails

Continued on next page

Exercise 5 Advanced – Reading & Writing to Database Tables, Continued

Part IV

Modify the transformation created in Part IV so that it has the following functionality:

- Writes the contents of the input database table to a text file according to the exercise detail below.

Exercise Detail	Value
Text file location	The parameterized output folder
Text file name	TextFileOutput.txt

End of Exercise

Congratulations! You have completed this exercise.

Guided Demo 8 – Data Cleansing

Introduction

In this guided demo, you will create a transformation that reads customer-related information from a database table. Most customers are assigned a sales rep, but several of the customer records do not. The transformation you build will find the customers that are missing an assigned sales rep and then output them to an Excel file.

Prerequisites

- The MySQL Server must be started.
- A connection named pentaho_oltp must have been created (done in a previous module).
- The DIR_OUTPUT parameter created in a previous module.

Objectives

After completing this guided demo, you will be able to:

- Filter the data stream on specific criteria.
- Sort records in the stream.
- Create an Excel .xls file containing data from the stream.

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Create the Transformation

Step	Action
1	To create the new transformation, press **CTRL-N**.
2	To save the Transformation, press **CTRL-S**.
3	Set the transformation properties for the **Transformation tab** according to the table below: **Property Name** / **Value** Transformation name: DataCleansing Description: Add a description of your choice. Directory: /public/PDI_Trn_Objects 🕮 NOTE: This is in the repository.
4	To close the 'Transformation properties' dialog, click **[OK]**
5	To customize the save comment, at the 'Enter a comment' dialog, **enter an optional comment**, and then, click **[OK]**.

Create the Table Input Step

In this section, you will create the transformation and add the table input step. You will define the step's name as well as the database connection to use when reading data.

Step	Action
1	To create the first step, from the **Input** category of the **Design** tab, drag the **Table Input** step onto the Canvas.
2	To open the step's properties dialog, double-click the step.
3	To set the step's name and database connection, set the properties according to the table below: **Property Name** / **Value** Step name: Get Customer Info Connection: pentaho_oltp

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Define the Table & Columns to Read

A table input step needs to know the table and which columns to read data from. It is defined in a SQL query. It is configured it in this section.

Step	Action
1	To open the Database Explorer, click **[Get SQL select statement…]**.
2	To expand the entire tree, click the **Expand/Collapse Tree** icon. 

3	To choose the table to be used for the query:
	
	In the Database Explorer, select the **customer table**.
	Click **[OK]**.
	At the 'Question?' dialog, click **[Yes]**.
	
	A SQL statement that will return all rows, for all columns for the selected table is automatically generated and entered in the SQL window of the Table input step's dialog. It should look like the screenshot below:
	
	`SELECT`
	`  customernumber`
	`, customername`
	`, contactlastname`
	`, contactfirstname`
	`, phone`
	`, addressline1`
	`, addressline2`
	`, city`
	`, state`
	`, postalcode`
	`, country`
	`, salesrepemployeenumber`
	`, creditlimit`
	`FROM customers`

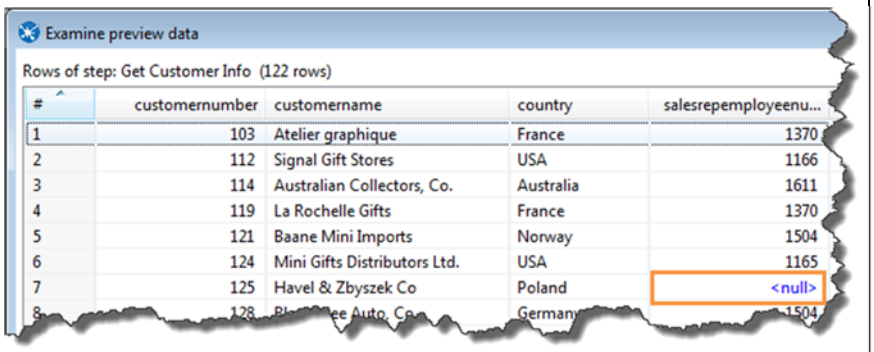
Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Define the Table & Columns to Read, continued

Step	Action

4	For this exercise, all columns are not necessary. Edit the SQL statement by removing all **except** the columns shown in the list below.
	
	customernumber
	customername
	country
	salesrepemployeenumber
	
	The finished SQL statement should look like screenshot below:
	
	`SELECT`
	`  customernumber`
	`, customername`
	`, country`
	`, salesrepemployeenumber`
	`FROM customers`

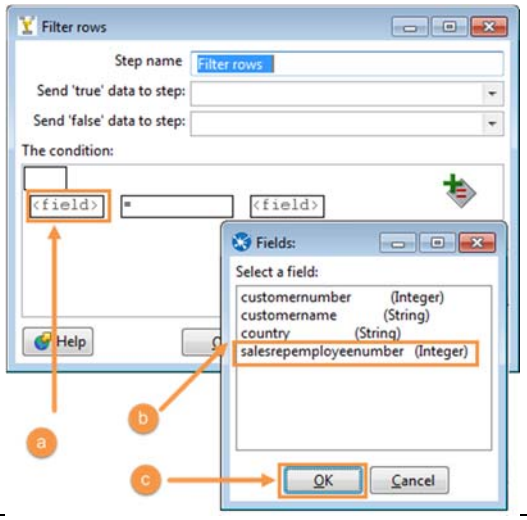
5	To verify the SQL is formatted correctly, click **[Preview]**, and then click **[OK]**.
6	The data returned should look like the example below.  Examine preview data Rows of step: Get Customer Info (122 rows) \# customernumber customername country salesrepemployeenu... 1 103 Atelier graphique France 1370 2 112 Signal Gift Stores USA 1166 3 114 Australian Collectors, Co. Australia 1611 4 119 La Rochelle Gifts France 1370 5 121 Baane Mini Imports Norway 1504 6 124 Mini Gifts Distributors Ltd. USA 1165 7 125 Havel & Zbyszek Co Poland <null> 8 128 ee Auto, Co. Germany 1504 🕮 NOTE: Notice that some of the customers contain a <null> for salesrepemployeenumber.
7	To close the 'Examine preview data' dialog, click **[Close]**.
8	To close the Table input step's property dialog, click **[OK]**.
9	To save the transformation, in the Spoon toolbar, click the **Save current file** icon.

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Create & Configure the Filter rows Step

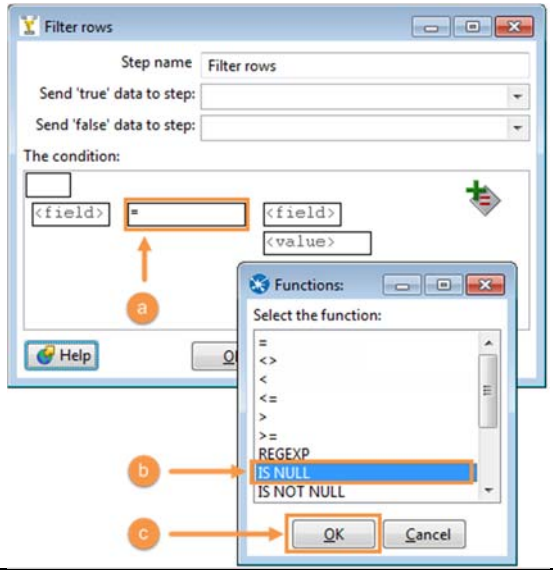
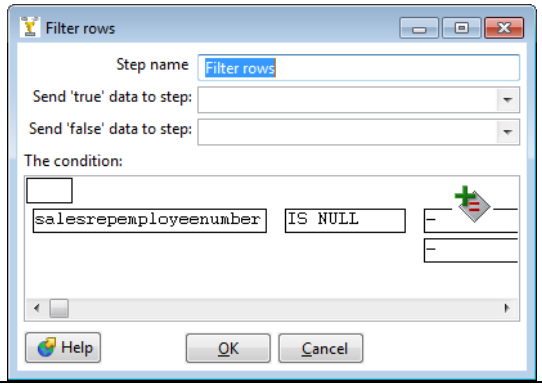
In this section, you add a Filter rows step that will separate out the rows in the stream that contain <null> for the salesrep.

Step	Action
1	To create the first step, from the **Flow** category of the **Design** tab, drag the **Filter rows** step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step** → **Destination Step** Table input → Filter rows
3	To open the step's properties dialog, double-click the step.
4	To configure the field that will be filtered on: Click in the **<field>** edit box. Select salesrepemployeenumber (Integer). Click **[OK]**. 

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Create & Configure the Filter rows Step, continued

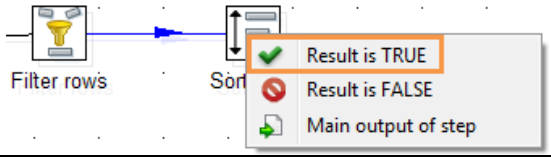
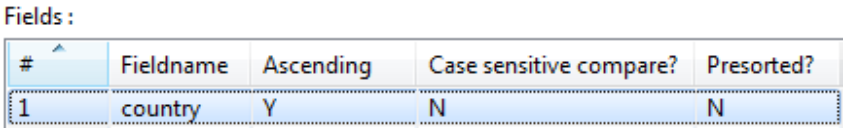
Step	Action
5	To configure the function (condition) to filter on: ■ Click the = edit box. ■ Select IS NULL. ■ Click **[OK]**. 
6	The completed step configuration should look like the screenshot below. 
7	To close the step's properties dialog, click **[OK]**.
8	**Save** the transformation.

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Create & Configure the Sort Rows Step

In this section of the exercise, you add a Sort rows step that sorts the stream by country in ascending order. The hop connecting the Filter rows and Sort row steps is configured to send all rows in the stream, where the filter rows condition is true, to the Sort rows step.

Step	Action
1	To create the next step, from the **Transform** category of the **Design** tab, drag the **Sort rows** step onto the Canvas.
2	Create a hop between the steps as shown in the table below, and then, select **Result is TRUE**. **Source Step** → **Destination Step**: Filter rows → Sort rows 
3	To open the step's properties dialog, **double-click the step**.
4	To configure the Fields grid, set it according to the screenshot shown below:  🕮 NOTE: All other properties on this tab should remain at their default values.
5	To close the step's properties dialog, click **[OK]**.
6	**Save** the transformation.

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Create & Configure the Excel Writer Step

Now that the data is sorted, you will add a step that will create an Excel .xls file and write the stream's data to it. A parameter (created in a previous module) is used to define the folder where the file is created.

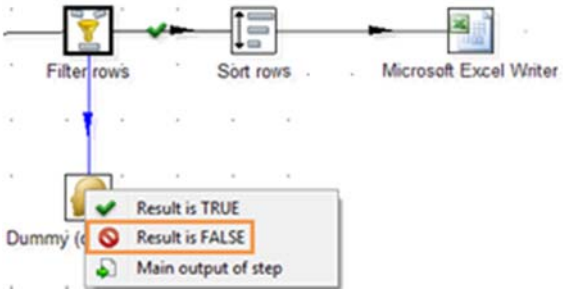
Step	Action
1	To create the first step, from the **Output** category of the **Design** tab, drag the **Microsoft Excel Writer** step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step** → **Destination Step** Sort rows → Microsoft Excel Writer
3	To open the step's properties dialog, double-click the step.
4	To configure the Filename and location where the file is created, in the **File & Sheet** tab, set the Filename, property according to the table shown below: **Property Name**: Filename **Value**: ${DIR_OUTPUT}\CustomersWoSalesreps 🕮 NOTE: It is not necessary to add the extension onto the filename. That is defined by default in the Extension property.
5	To configure the fields to include in the output file: Click the **Content** tab. Click the **[Get Fields]** button.
6	To close the step's properties dialog, click **[OK]**.
7	Save the transformation.

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Create & Configure the Dummy Step

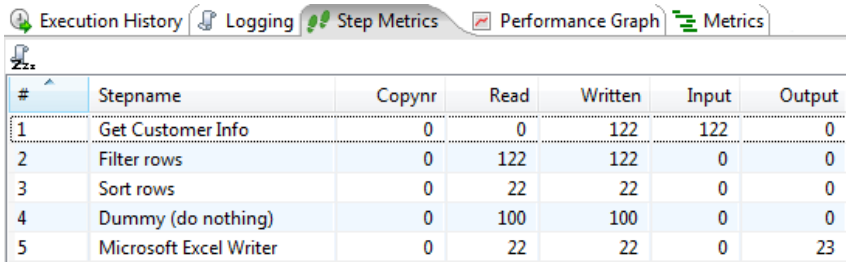
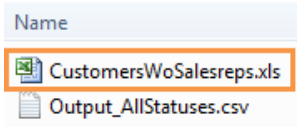
To finish creating the transformation, you will add a Dummy (do nothing) step, and create a hop to it from Sort rows. The hop will be configured to send any data in the stream that fails the condition defined in the Sort rows step, to the Dummy step. This is done because we are not interested in keeping that information for this transformation. It still resides in the table it was read from.

Step	Action
7	To create the Dummy step, from the **Flow** category of the **Design** tab, drag the **Dummy (do nothing)** step onto the Canvas, placing it below the Filter rows step.
8	Create a hop between the steps as shown in the table below, and then, select **Result is FALSE**. **Source Step** | **Destination Step** Sort rows | Dummy (do nothing)  📖 NOTE: Notice the how PDI places indicators on the hops to show which hop is for True ✔ and which one is for False. ⛔
9	Save the transformation.

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

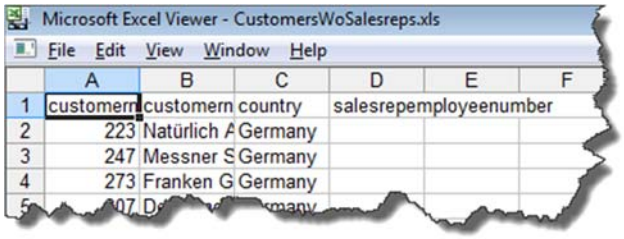
Execute the Transformation

Step	Action
1	To run the transformation, press **F9**, and then, click **[Launch]**.
2	There should be no errors and the **Step Metrics** tab in the Execution Results pane should look similar to the screenshot shown below:  Execution History | Logging | Step Metrics | Performance Graph | Metrics | # | Stepname | Copynr | Read | Written | Input | Output | |---|---|---|---|---|---|---| | 1 | Get Customer Info | 0 | 0 | 122 | 122 | 0 | | 2 | Filter rows | 0 | 122 | 122 | 0 | 0 | | 3 | Sort rows | 0 | 22 | 22 | 0 | 0 | | 4 | Dummy (do nothing) | 0 | 100 | 100 | 0 | 0 | | 5 | Microsoft Excel Writer | 0 | 22 | 22 | 0 | 23 | 🕮 NOTE: Notice the Output field contains data. This is indicates the number of lines output to the Excel file that it created.
3	To verify the Excel file was created, navigate to the folder shown below and verify the creation of the file as show in in the example screenshot: **C:\pentahotraining\DataFiles\Output**  Name CustomersWoSalesreps.xls Output_AllStatuses.csv

Continued on next page

Guided Demo 8 – Data Cleansing, Continued

Execute the Transformation, continued

Step	Action
4	Open the xls file by double-clicking it and notice how orders from all of the input text files are included. Example of **CustomersWoSalesreps.xls**  🕮 NOTE: Notice how there are no sales reps, just as it was designed.

Solution Details

The solution to this exercise can be obtained using the details below:

Location: `C:\pentahotraining\Solutions\Guided Demos`

Completed transformation:

`GD8_DataCleansing.ktr`
(Use **File | Import from an XML file** to import)

Text file output: `CustomersWoSalesreps.xls`

End of Exercise

Congratulations! You have completed this exercise.

Exercise 6 – Input with Parameters & Table Copy Wizard

Introduction

In this exercise, you will use the Insert/Update step. Create a transformation that uses parameters, and use the Copy Table wizard tool.

📖 NOTE

For those looking for more of a challenge, try the advanced version of this exercise. It is the same exercise, but without the detailed guidance. You will find it in this workbook immediately following this exercise.

Objectives

After completing this exercise, you will be able to:

- Create a transformation that uses the CSV file input and Insert/Update steps
- Create a transformation that uses the Table input step that loads data based on a parameter value
- Use the Copy Table wizard

Prerequisites

You must complete the prior exercise on installing and configuring Pentaho Data Integration in order to complete this exercise. You must also have access to course files required (if any).

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard, Continued

Part I: CSV file input and Insert/Update

In part I of this exercise you create a transformation that uses the CSV file input and Insert/Update steps to capture updated data in the source file.

Using CSV file input and Insert/Update steps

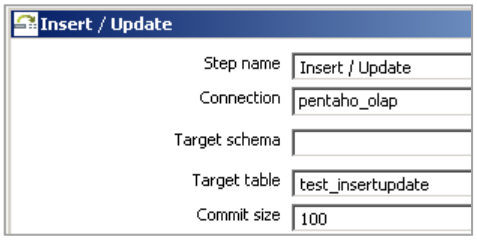
To create a transformation that uses CSV file input and Insert/Update steps:

Step	Action
1	Create a new transformation.
2	Save the file in `C:\pentahotraining\My Work\EX6\` using **File name**: `CSVFileInput_InsertUpdate`.
3	Drag an **Input > CSV file input** step onto the canvas.
4	Double-click **CSV file input**.
5	In the 'CSV input' dialog, type (or browse to) the following: **Field** / **Value** Filename: C:\pentahotraining\DataFiles\Output\TextFileOutput.txt Delimiter: ;
6	Click **Get Fields**.
7	In the 'Sample size' dialog, accept the default value (**100**) and click **[OK]**.
8	When you receive the **Scan results** dialog, click **Close**.
9	The fields table should look like the following (depending on your locale, the Currency, Decimal and Group characters may vary): Name / Type / Format / Length / Precision / Currency / Decimal / Group ordernumber / Integer / # / 15 / / $ / . / , productcode / String / / 9 / / $ / . / , quantityordered / Integer / # / 15 / / $ / . / , priceeach / Number / #.# / 15 / / $ / . / , total / Number / #.# / 15 / / $ / . / , orderlinenumber / Integer / # / 15 / / $ / . / ,
10	Click **[OK]** to close the 'CSV Input' dialog.
11	Drag an **Output > Insert/Update** step onto the canvas.
12	Create a **hop** between **CSV file input** and **Insert/Update**.
13	If you receive a pop-up dialog creating the hop, choose **Main output of step**.
14	Double-click **Insert/Update**.

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard,
Continued

Using CSV file input and Insert/Update steps (continued)

Step	Action
15	In the 'Insert/Update' dialog, type or choose the following: **Field** / **Value** Connection: **pentaho_olap** Target schema: **[leave blank]** Target table: **test_insertupdate** Commit size: **100** 
16	To the right of the "The key(s) to look up the value(s)" table, click **Get Fields**.
17	By selecting the row and pressing **Delete**, remove all fields from **The key(s) to look up the value(s)** table except the two key values: **ordernumber** and **orderlinenumber**. The key(s) to look up the value(s): #. | Table field | Comparator | Stream field1 1 | ordernumber | = | ordernumber 2 | orderlinenumber | = | orderlinenumber
18	To the right of the **Update fields** table, click **Get update fields**. Do not remove any fields from the results. Update fields: #. | Table field | Stream field | Update 1 | ordernumber | ordernumber | Y 2 | productcode | productcode | Y 3 | quantityordered | quantityordered | Y 4 | priceeach | priceeach | Y 5 | total | total | Y 6 | orderlinenumber | orderlinenumber | Y

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard,
Continued

Using CSV file input and Insert/Update steps
(continued)

Step	Action

19	Click the **SQL** button. The SQL code returned should look like the following:
	`CREATE TABLE test_insertupdate`
	`(`
	`  ordernumber BIGINT`
	`, orderlinenumber BIGINT`
	`, productcode VARCHAR(9)`
	`, quantityordered BIGINT`
	`, priceeach DOUBLE`
	`, total DOUBLE`
	`)`
	`;CREATE INDEX idx_test_insertupdate_lookup`
	`ON test_insertupdate`
	`(`
	`  ordernumber`
	`, orderlinenumber`
	`)`
	`;`

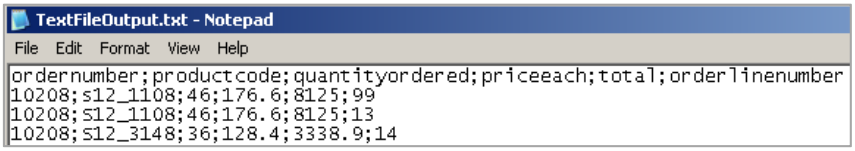
20	In the **Simple SQL editor**, click **Execute**.
21	Click **[OK]** to close the results dialog.
22	Close the Simple SQL editor.
23	Click **[OK]** to close the 'Insert / Update' dialog.
24	Save your work.
25	Click the **Run this transformation** icon, then click **Launch**.
26	Switch to the **Step Metrics** view and note the **Output** of the **Insert/Update** step (**23640** rows). Execution History | Logging | Step Metrics | Performance Graph #. Stepname Copynr Read Written Input Output 1 CSV file input 0 0 23640 23641 0 2 Insert / Update 0 23640 23640 23640 23640
27	Run the transformation again. In the **Step Metrics** view, note the **Output** of the **Insert/Update** step is now **0**. Execution History | Logging | Step Metrics | Performance Graph #. Stepname Copynr Read Written Input Output 1 CSV file input 0 0 23640 23641 0 2 Insert / Update 0 23640 23640 23640 0

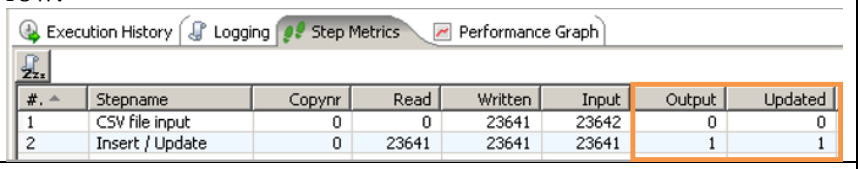
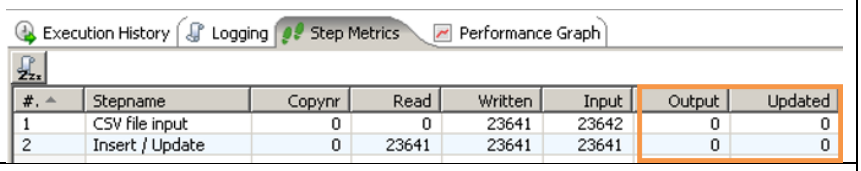
Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard, Continued

Using CSV file input and Insert/Update steps (continued)

Step	Action
28	Open C:\pentahotraining\Data Files\Output\TextFileOutput.txt in a text editor.

29	Add the following line as the first row below the headers:
	`10208;s12_1108;46;176.6;8125;99`
	And change the quantityordered field of the third entry in the file from 26 to **36**:
	`10208;S12_3148;36;128.4;3338.9;14`
	 

30	Close the file and save your changes.
31	Click the **Run this transformation** icon, then click **Launch**.
32	In the **Step Metrics** view, note that the output of the **Insert/Update** step shows **1** new **Output** row and **1 Updated** row. 
33	Run the transformation again. The output for the **Insert/Update** step should now show **0 Output** and **0 Updated**. 
34	Close "CSVFileInput_InsertUpdate.ktr."

Part II: Table input with parameters

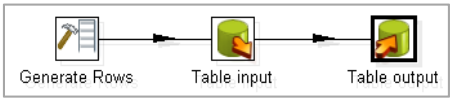
In part II of this exercise you create a transformation that uses the Table input step to load data based on a parameter value. This procedure is commonly used to load only changed (or delta) data into a data warehouse.

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard, Continued

Using a Table input step with parameters

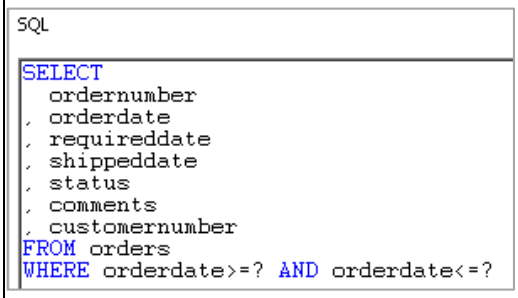
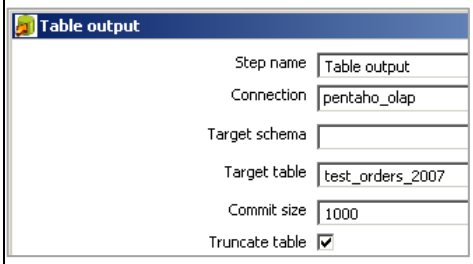
To create a transformation that uses the Table input step with parameters:

Step	Action
1	Create and save a new transformation named: `C:\pentahotraining\My Work\EX6\TableInputParameter`.
2	Drag an **Input** > **Generate Rows** and an **Input** > **Table input** step onto the canvas.
3	Drag an **Output** > **Table output** step onto the canvas.
4	Create a **hop** between **Generate Rows** and **Table input**.
5	Create a **hop** between **Table input** and **Table output**. 
6	Save your work.
7	Double-click the **Generate Rows** step.
8	In the **Generate Rows** dialog, change the **Limit** to **1**.
9	In the **Fields** table, type or choose the following: **Name** / **Type** / **Format** / **Value** OrderDateFrom / Date / yyyy-MM-dd / 2007-01-01 OrderDateTo / Date / yyyy-MM-dd / 2007-12-31 Name | Type | Format | Length | Precision | Currency | Decimal | Group | Value OrderDateFrom | Date | yyyy-MM-dd | | | | | | 2007-01-01 OrderDateTo | Date | yyyy-MM-dd | | | | | | 2007-12-31
10	Click **[OK]** to close the 'Generate Rows' dialog.
11	Double-click the **Table Input** step.
12	In the 'Table input' dialog, for **Connection**, select **pentaho_oltp**.
13	Click **Get SQL select statement**.
14	In **Database Explorer**, expand **pentaho_oltp** > **Tables**.
15	Select **orders** and click **[OK]**.
16	When prompted, click **Yes** to include the field names in the SQL statement.

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard, Continued, Continued

Using a Table input step with parameters (continued)

Step	Action
17	Add the following additional line to the end of the SQL statement: `WHERE orderdate>=? AND orderdate<=?` (The '**?**' represents the parameterized value). 
18	For **Insert data from step**, choose **Generate Rows** from the drop-down list.
19	Click **[OK]** to close the 'Table input' dialog.
20	Double-click the **Table output** step.
21	In the 'Table output' dialog, type or choose the following: **Field** / **Value** Connection: **pentaho_olap** Target schema: **[leave blank]** Target table: **test_orders_2007** Commit size: **1000** Truncate table: **[checked]** 
22	Click the [**SQL**] button.

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard, Continued

Using a Table input step with parameters (continued)

Step	Action

23	The following SQL should appear in the **Simple SQL editor**:
	`CREATE TABLE test_orders_2007`
	`(`
	`  ordernumber INT`
	`, orderdate DATETIME`
	`, requireddate DATETIME`
	`, shippeddate DATETIME`
	`, status VARCHAR(15)`
	`, comments TEXT`
	`, customernumber INT`
	`)`
	`;`
	Click **Execute** to run the SQL code.

24	When you receive the 'Results' dialog, click **[OK]**.
25	Close the "Simple SQL editor."
26	Click **[OK]** to close the "Table output" dialog.
27	Save your work.
28	Click the **Run this transformation** icon, then click **Launch**.
29	In the **Step Metrics** view, note **320** rows **Written** by **Table input** and **320** rows **Read** and **Output** by the **Table output** step. Execution History | Logging | Step Metrics | Performance Graph #. Stepname Copynr Read Written Input Output 1 Generate Rows 0 0 1 0 0 2 Table input 0 1 320 320 0 3 Table output 0 320 320 0 320
30	Switch to the **View** tab.
31	Expand **Transformations** > **TableInputParameter** > **Database connections**.
32	Right-click **pentaho_olap** and choose **Explore** from the menu.
33	In **Database Explorer**, expand **pentaho_olap** > **Tables**.
34	Right-click **test_orders_2007** and choose **Preview first 100**. Rows of step: test_orders_2007 (100 rows) #. ordernumber orderdate requireddate shippeddate 1 70510100 2007/01/11 00:00:00.000 2007/01/18 00:00:00.000 2007/01/15 00:00:00.000 2 70510101 2007/01/14 00:00:00.000 2007/01/23 00:00:00.000 2007/01/16 00:00:00.000 3 70510102 2007/01/15 00:00:00.000 2007/01/23 00:00:00.000 2007/01/19 00:00:00.000 4 70510103 2007/02/03 00:00:00.000 2007/02/12 00:00:00.000 2007/02/07 00:00:00.000 5 70510104 2007/02/05 00:00:00.000 2007/02/14 00:00:00.000 2007/02/06 00:00:00.000 6 70510105 2007/02/16 00:00:00.000 2007/02/26 00:00:00.000 2007/02/17 00:00:00.000 7 70510106 2007/02/22 00:00:00.000 2007/03/01 00:00:00.000 2007/02/26 00:00:00.000 8 70510107 2007/03/01 00:00:00.000 2007/03/08 00:00:00.000 2007/03/02 00:00:00.000 9 70510108 2007/03/08 00:00:00.000 2007/03/17 00:00:00.000 2007/03/13 00:00:00.000 10 70510109 2007/03/15 00:00:00.000 2007/03/24 00:00:00.000 2007/03/16 00:00:00.000

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard, Continued

Using a Table input step with parameters (continued)

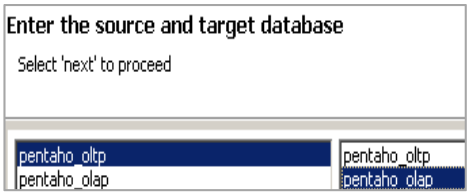
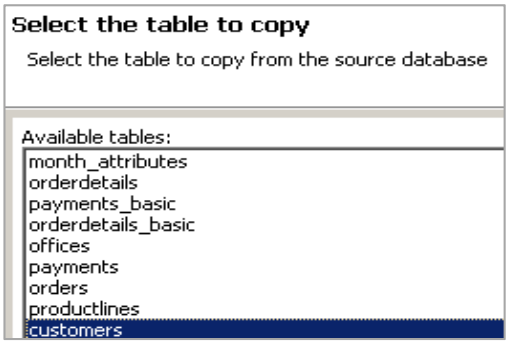
Step	Action
35	Close the preview data window.
36	Click **[OK]** to close Database Explorer.
37	Close "TableInputParameter.ktr."

Part III: Copy Table wizard

In optional part III of this exercise, you use the Copy Table wizard to move data from one database to another.

Using the Copy Table wizard

To use the Copy Table wizard:

Step	Action
1	Create a new transformation (you will save it later).
2	From the menu options choose **Tools | Wizard | Copy table** (*not* Copy Tables).
3	At the **Enter the source and target database** panel, in the left pane select **pentaho_oltp**. In the right panel, select **pentaho_olap**. 
4	Click **Next**.
5	At the **Select the table to copy** panel, select **customers**. 

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard, Continued

Using the Copy Table wizard (continued)

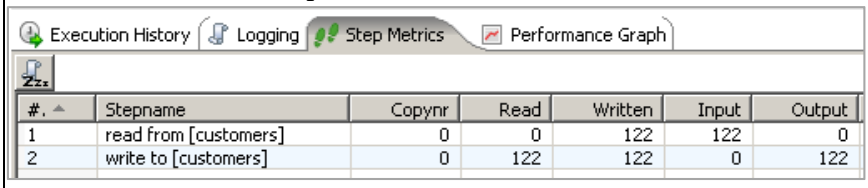
Step	Action
6	Click **Finish**.
7	Save the transformation in `\My Work\EX6\` using **File name**: `CopyTableWizard`.
8	Double-click the **write to [customers]** step.
9	In the 'Table output' dialog, click the **SQL** button.

10	The "Simple SQL editor" should display the code necessary to create the customers table in the pentaho_olap database:
	`CREATE TABLE customers`
	`(`
	`  customernumber INT`
	`, customername VARCHAR(50)`
	`, contactlastname VARCHAR(50)`
	`, contactfirstname VARCHAR(50)`
	`, phone VARCHAR(50)`
	`, addressline1 VARCHAR(50)`
	`, addressline2 VARCHAR(50)`
	`, city VARCHAR(50)`
	`, state VARCHAR(50)`
	`, postalcode VARCHAR(15)`
	`, country VARCHAR(50)`
	`, salesrepemployeenumber INT`
	`, creditlimit DOUBLE`
	`)`
	`;`

Continued on next page

Exercise 6 – Input with Parameters & Table Copy Wizard, Continued, Continued

Using the Copy Table wizard, continued

Step	Action
11	Click **Execute**.
12	Click **[OK]** to close the 'Results' dialog.
13	Close the 'Simple SQL editor.'
14	Click **[OK]** to close the 'Table output' dialog.
15	Save your work.
16	Click the **Run this transformation** icon, then click **Launch**. 122 rows should be copied from one table to the other. 

#.	Stepname	Copynr	Read	Written	Input	Output
1	read from [customers]	0	0	122	122	0
2	write to [customers]	0	122	122	0	122

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Exercises`

Completed transformations:
`EX6_CSVFileInput_InsertUpdate.ktr`
`EX6_TableInputParameter.ktr`
`EX6_CopyTableWizard.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Exercise 6 Advanced – Input with Parameters & Table Copy Wizard

Introduction

This is an advanced version of Exercise 6. It is the same exercise, but without the detailed guidance. You may choose to do this advanced exercise rather than the standard version of Exercise 6.

Prerequisites

You must have the 'pentaho_olap' and 'pentaho_oltp' database connections. You must also have access to pentahotraining files required (if any).

Instructions & Objectives

Please read each part section below to see the requirements and instructions necessary for completing each part this advanced exercise.

After completing all parts of this exercise, you will be able to:

- Create a transformation that uses the Generate Rows step as a parameter for loading a table with data. Choose any database table you like.
- Use the Copy Table wizard to copy contents of the customers table from pentaho_oltp to pentaho_olap.

End of Exercise

Congratulations! You have completed this exercise.

Exercise 7 – Parallel Processing

Introduction

In this exercise, you will learn how to create and then execute more than one instance of a step at the same time.

Prerequisites

None.

Objectives

After completing this exercise, you will be able to:

- Copy and Distribute data
- Start multiple copies of a step.
- Send data to multiple outputs

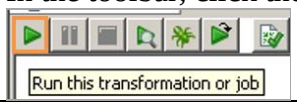
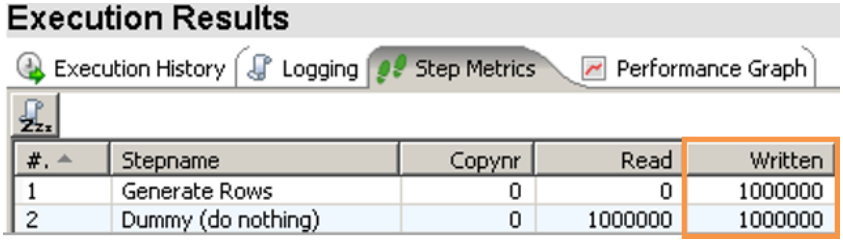
Part I: Understanding Data Flow

In this exercise you will start multiple copies of a step and distribute the data to multiple outputs.

Continued on next page

Exercise 7 – Parallel Processing, Continued

Understanding Data Flow

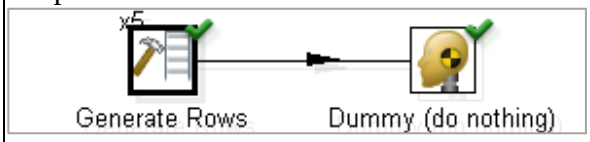
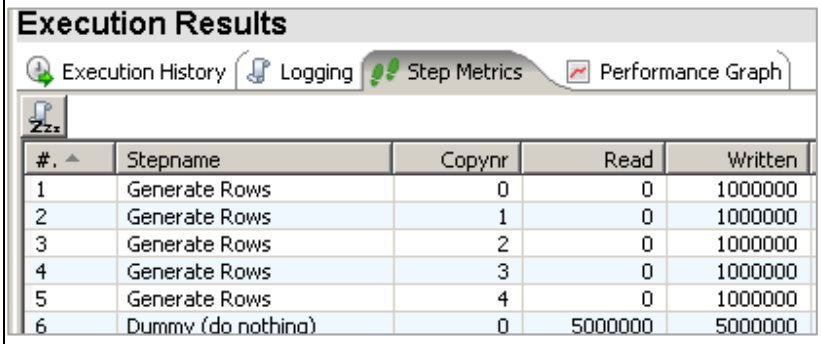
Step	Action
1	Open the transformaion, **HelloWorld**.
2	Double-click the **Generate Rows** step, change the **Limit** field to **1000000** (one million), and click **[OK]**.
3	Save the transformation as **HelloWorld_ParallelProcessing**.
4	In the toolbar, click the **Run this transformation or job** button. 
5	In the 'Execute a transformation' dialog, accept the default options and click **Launch**.
6	Switch to the **Step Metrics** view and note the number of rows written equals 1,000,000. 

#.	Stepname	Copynr	Read	Written
1	Generate Rows	0	0	1000000
2	Dummy (do nothing)	0	1000000	1000000

Continued on next page

Exercise 7 – Parallel Processing, Continued

Part I: Understanding Data Flow (continued)

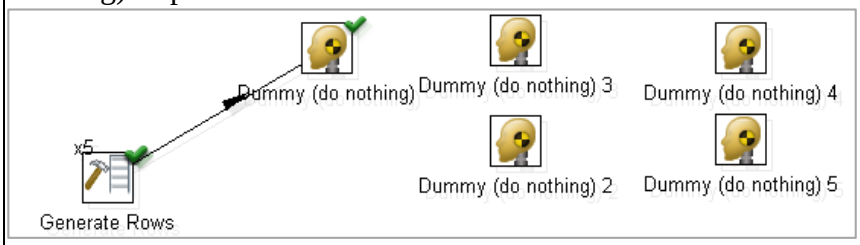
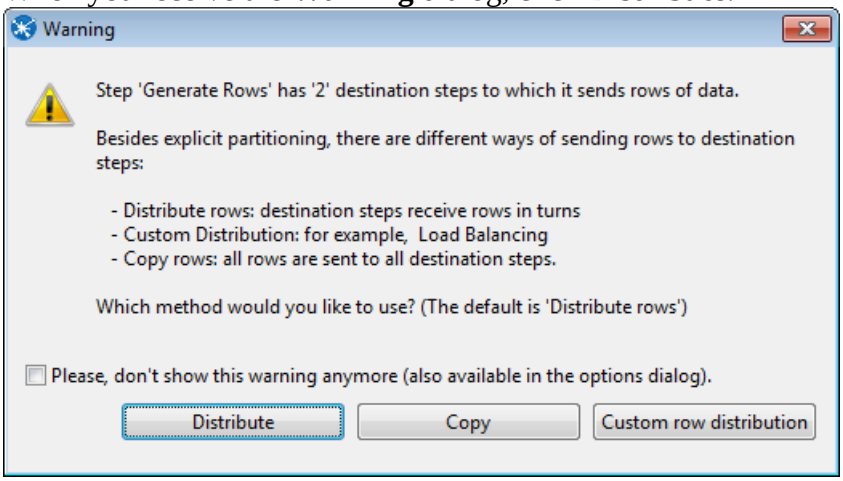
Step	Action
8	Right-click the **Generate Rows** step and select **Change the number of copies to start** from the context menu.
9	In the 'Nr of copies of step' dialog, change the **Number of copies** field to **5**, and click **[OK]**. This adds a X5 marker to the step on the canvas. 
10	Save the transformation as **HelloWorld_ParallelProcessing_5copies**.
11	Click the **Run this transformation or job** button.
12	In the 'Execute a transformation' dialog, click **Launch**.
13	Switch to the **Step Metrics** view. Notice the transformation wrote 5x1,000,000 rows by spawning 5 copies of the Generate Rows step. Also notice the Dummy step reads 5,000,000 rows. 

#.	Stepname	Copynr	Read	Written
1	Generate Rows	0	0	1000000
2	Generate Rows	1	0	1000000
3	Generate Rows	2	0	1000000
4	Generate Rows	3	0	1000000
5	Generate Rows	4	0	1000000
6	Dummy (do nothing)	0	5000000	5000000

Continued on next page

Exercise 7 – Parallel Processing, Continued

Part I: Understanding Data Flow (continued)

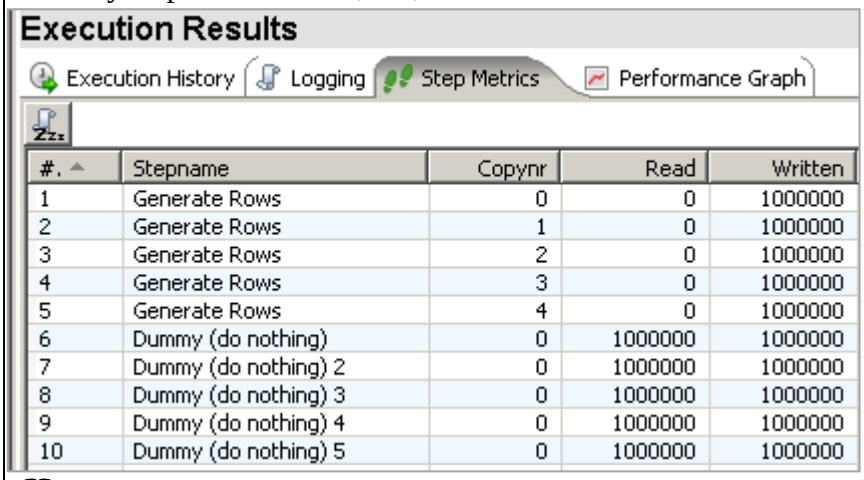
Step	Action
14	Using **Edit | Copy** and **Paste**, create a total of **5** Dummy (do nothing) steps on the canvas. 
15	Save your work as: **HelloWorld_ParallelProcessing_Distribute.**
16	Select **Generate Rows**, hold the **Shift** key and click and drag onto **Dummy (do nothing) 2**. This will create a hop between the two steps.
17	When you receive the **Warning** dialog, click **Distribute**. 
18	Using the previous steps (or the context menu), create hops between **Generate Rows** and the remaining **Dummy** steps.
19	**Save your work** or you will not see the expected results. .
20	Click the **Run this transformation or job** button.
21	In the 'Execute a transformation' dialog, click **Launch**.

Continued on next page

Exercise 7 – Parallel Processing, Continued

Part I: Understanding Data Flow (continued)

Step	Action
22	Switch to the **Step Metrics** view. Note that each of the 5 Dummy steps now reads 1,000,000 rows.



#.	Stepname	Copynr	Read	Written
1	Generate Rows	0	0	1000000
2	Generate Rows	1	0	1000000
3	Generate Rows	2	0	1000000
4	Generate Rows	3	0	1000000
5	Generate Rows	4	0	1000000
6	Dummy (do nothing)	0	1000000	1000000
7	Dummy (do nothing) 2	0	1000000	1000000
8	Dummy (do nothing) 3	0	1000000	1000000
9	Dummy (do nothing) 4	0	1000000	1000000
10	Dummy (do nothing) 5	0	1000000	1000000

	🕮 NOTE: This is the same result as right-clicking the Dummy (do nothing) step and changing "Change the number of copies to start" to 5.
23	Close the "First steps" transformation.

Part II: Parallelism

In part III of this exercise, you create and run a transformation that implements parallel processing.

Working with parallel processing

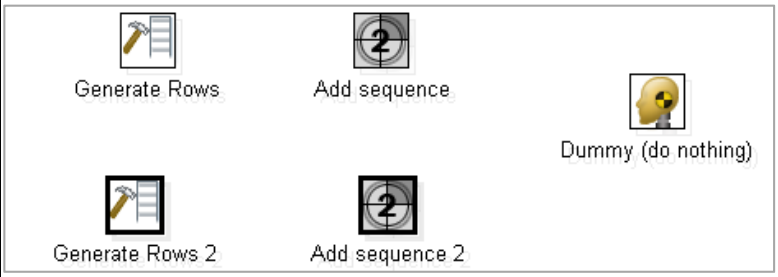
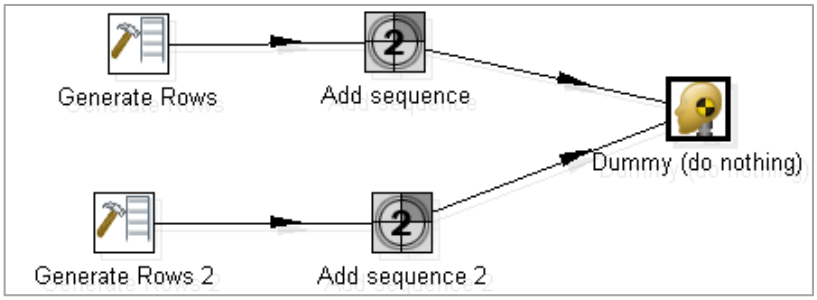
To implement parallel processing:

Step	Action
1	Choose **File** | **New** | **Transformation** from the menu options.
2	Choose **File** | **Save**.
3	Save the transformation using File name: **EX7_ParalellProcessing_DataFlow**.
4	Expand **Input**, then click and drag 2 **Generate Rows** steps onto the canvas.
5	Expand **Transform**, then click and drag 2 **Add Sequence** steps onto the canvas.

Continued on next page

Exercise 7 – Parallel Processing, Continued

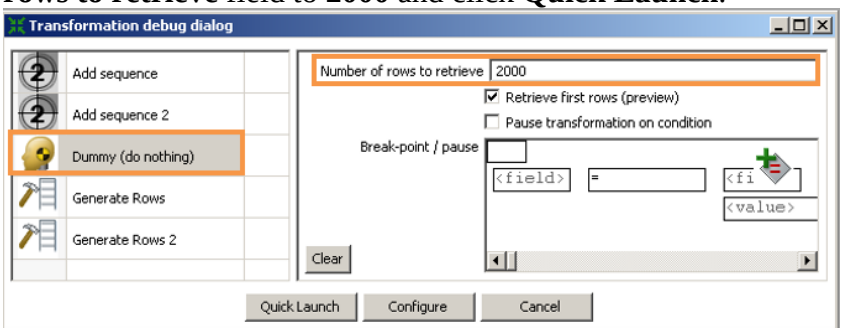
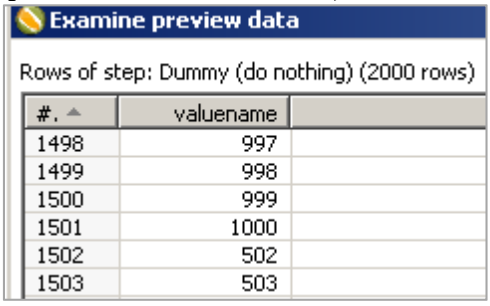
Working with parallel processing (continued)

Step	Action
6	Expand **Flow**, then click and drag a **Dummy (do nothing)** step onto the canvas. 
7	Hover the mouse pointer over the first **Generate Rows** step.
8	Add a hop.
9	Click the **Add sequence** step to create a **hop** between **Generate Rows** and **Add sequence**.
10	Repeat the previous steps to add a **hop** between **Generate Rows 2** and **Add sequence 2**.
11	Repeat the previous steps to add a **hop** between **Add sequence** and **Dummy**.
12	Repeat the previous steps to add a **hop** between **Add sequence 2** and **Dummy**. 

Continued on next page

Exercise 7 – Parallel Processing, Continued

Working with parallel processing (continued)

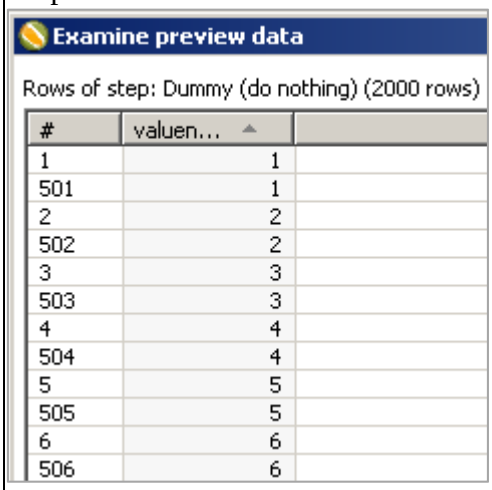
Step	Action
13	Edit each **Generate Rows** step and change the **Limit** (number of rows) to **1000**. You do not need to add fields.
14	Double-click **Add sequence** to edit it.
15	In the **Counter name (optional)** field, type: **CounterA**.
16	Edit **Add sequence 2** and change the **Counter name (optional)** to: **CounterB**.
17	Save your work.
18	Select the **Dummy** step, right-click, and choose **Preview** from the context menu.
19	In the 'Transformation debug' dialog, change the **Number of rows to retrieve** field to **2000** and click **Quick Launch**.  📖 NOTE: Be sure the **Dummy step is highlighted** in the Transformation debug dialog or the results received will be incorrect.
20	In the 'Examine preview data' dialog, note that the entries in the **valuename** column are out of sequence. (You may need to scroll past row # 500 to see this). 

#.	valuename
1498	997
1499	998
1500	999
1501	1000
1502	502
1503	503

Continued on next page

Exercise 7 – Parallel Processing, Continued

Working with parallel processing (continued)

Step	Action
21	Click the **valuename** column header to sort the values. Note the duplicate values. 
22	Close the **Examine preview data** window and close the **Select the preview step** dialog.
23	Edit the **Add sequence** and **Add sequence 2** steps and change **Counter name (optional)** for both steps to **lab2**.
24	Save your work.

Continued on next page

Exercise 7 – Parallel Processing, Continued, Continued

Working with parallel processing, continued

Step	Action
25	Select the **Dummy** step, right-click, and choose **Preview** from the context menu.
26	In the **Transformation debug** dialog, change the **Number of rows to retrieve** field to **2000** and click **Quick Launch**. 🕮 NOTE: Be sure the Dummy step is highlighted in the Transformation debug dialog or the results received will be incorrect.
27	Click the **valuename** column header to sort the values. Note the values are now consecutive: 1-2000.
28	Close the "Examine preview data" dialog, close the "Select the preview step" dialog, and close the Data flow transformation.

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Exercises`

Completed transformations:
`EX7_HelloWorld_ParallelProcessing_5copies.ktr`
`EX7_HelloWorld_ParallelProcessing_Distribute.ktr`
`EX7_ParalellProcessing_DataFlow.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields'

Introduction

In this guided demo, you will witness the importance of choosing an adequate number of sample rows to enter when clicking the [Get Fields] button for input steps. You will be guided to purposely choose a number of sample rows that is not enough for PDI to correctly determine the correct column length for a CSV file input step's fields. Then, you will attempt to load the data into a table whose column lengths were determined from the CSV file input step's fields. This results in an error during the table load.

📖 NOTE: This guided demo is performed at this stage in the course due to its proximity to the exercises where the importance of the correct preview sample size is paramount.

Objectives

After completing this guided demo, you will be able to:

- Properly choose a sample size for a CSV file input step. The same technique is used for a Text file input step as well.
- Identify transformation errors whose cause may be an improper sample size chosen in a previous step.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

Create the Transformation

Step	Action
1	To create the new transformation, press **CTRL-N**.
2	To save the Transformation, press **CTRL-S**.
3	Set the transformation properties for the **Transformation tab** according to the table below: **Property Name** / **Value** Transformation name: GetFieldsSampleSize Description: Add a description of your choice. Directory: /public/PDI_Trn_Objects 📖 NOTE: This is in the repository.
4	To close the 'Transformation properties' dialog, click **[OK]**.
5	To customize the save comment, at the 'Enter a comment' dialog, **enter an optional comment**, and then, click **[OK]**.

Create & Configure the CSV File Input Step

In this portion of the guided demo, you will create a new transformation and add a CSV file input step. Then you will configure it read the sales_data.csv file that comes with every PDI install as part of its sample data.

Step	Action
1	To create the first step, from the **Input** category of the **Design** tab, drag the **CSV file output** step onto the Canvas.
2	To open the step's properties dialog, double-click the step.

Continued on next page

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

Create & Configure the CSV File Input Step, continued

Step	Action
3	To configure the Filename property, set it according to the table shown below: **Property Name**: Filename **Value**: C:\Pentaho\design-tools\data-integration\samples\transformations\files\sales_data.csv

Demonstrate Proper Sample Size

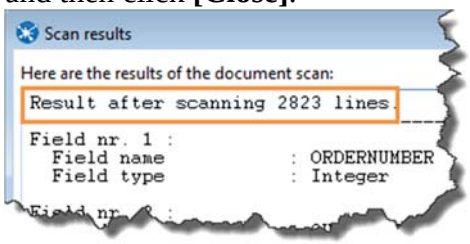
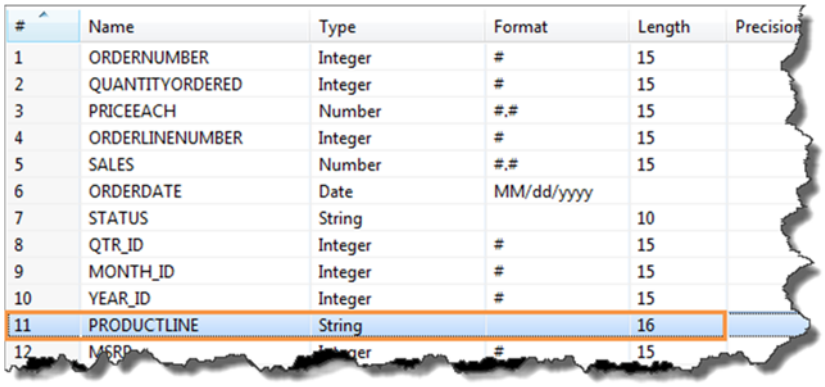
In this section of the guided demonstration, you will configure the fields for the CSV file input step to read the entire file to determine the proper field lengths.

Step	Action
1	To have PDI read the entire contents of the file to determine the proper field lengths, and configure the fields: ■ In the 'CSV Input' dialog, click **[Get Fields]**. ■ In the 'Sample size' dialog, enter 0 (zero), and then click **[OK]**.

Continued on next page

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

Demonstrate Proper Sample Size, continued

Step	Action
2	Notice in the 'Scan results' dialog that PDI scanned 2823 lines, and then click **[Close]**. 
3	Examine the field lengths that PDI has automatically configured for you. Specifically, notice the field named PRODUCTLINE. Its length is correctly set to 16. 

#	Name	Type	Format	Length	Precision
1	ORDERNUMBER	Integer	#	15	
2	QUANTITYORDERED	Integer	#	15	
3	PRICEEACH	Number	#.#	15	
4	ORDERLINENUMBER	Integer	#	15	
5	SALES	Number	#.#	15	
6	ORDERDATE	Date	MM/dd/yyyy		
7	STATUS	String		10	
8	QTR_ID	Integer	#	15	
9	MONTH_ID	Integer	#	15	
10	YEAR_ID	Integer	#	15	
11	PRODUCTLINE	String		16	
12	MSRP	Integer	#	15	

CAUTION

If you are reading a very large file, it may take a long time to read all of its contents. In that case, it would be better to determine beforehand the proper field lengths or calculate a sample size that is smaller than the entire file, but large enough to encapsulate the largest field values.

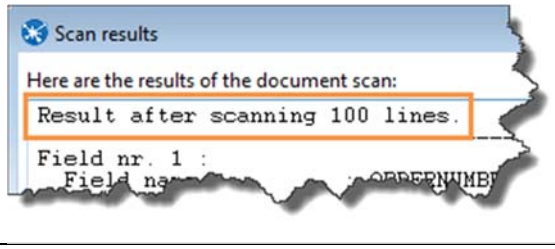
Continued on next page

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

Demonstrate Using an Improper Sample Size

In this section of guided demonstration, you will configure the fields for the CSV file input step again, but this time using an inadequate sample size. To demonstrate how a transformation can behave when the field lengths are not correct, you will continue to build the transformation by creating a table to load the data into using a table output step. The length of the PRODUCTLINE field in the table created will not be large enough to contain all of the data for all of the rows in the CSV file.

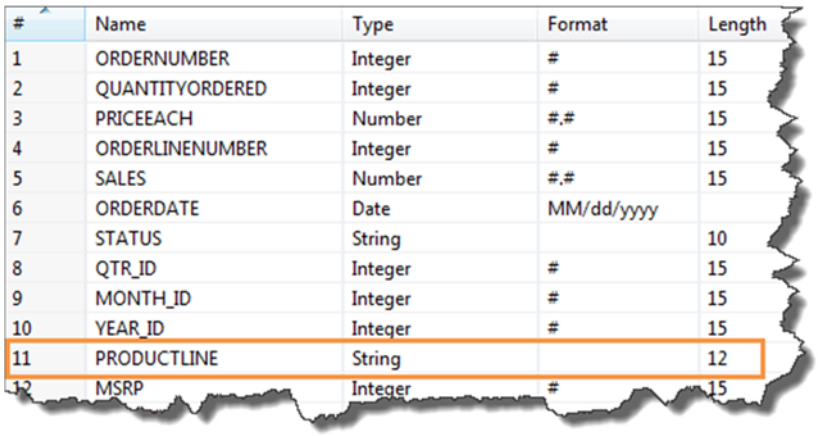
You will witness an error when the transformation is run and the table load is attempted. This is expected.

Step	Action
1	To have PDI read the entire contents of the file to determine the proper field lengths, and configure the fields: ■ In the 'CSV Input' dialog, click **[Get Fields]**. ■ In the 'Sample size' dialog, enter 100, and then click **[OK]**. 🕮 NOTE**:** 100 is PDI's default sample size. Notice in the 'Scan results' dialog that PDI scanned 100 lines, and then click **[Close]**. 

Continued on next page

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

Demonstrate Using an Improper Sample Size, continued

Step	Action
2	Examine the field lengths that PDI has automatically configured for you. Specifically, notice the field named PRODUCTLINE. Its length is set to 12. You now know that 12 is too small and not all data for that row in the CSV file can fit into the field.  📖 NOTE**:** There are several fields that are problematic using this sample size. For the purpose of this guided demonstration, we are focusing on the PRODUCTLINE field.
3	To close the 'CSV Input' dialog, click **[OK]**.

#	Name	Type	Format	Length
1	ORDERNUMBER	Integer	#	15
2	QUANTITYORDERED	Integer	#	15
3	PRICEEACH	Number	#,#	15
4	ORDERLINENUMBER	Integer	#	15
5	SALES	Number	#,#	15
6	ORDERDATE	Date	MM/dd/yyyy	
7	STATUS	String		10
8	QTR_ID	Integer	#	15
9	MONTH_ID	Integer	#	15
10	YEAR_ID	Integer	#	15
11	PRODUCTLINE	String		12
12	MSRP	Integer	#	15

Continued on next page

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

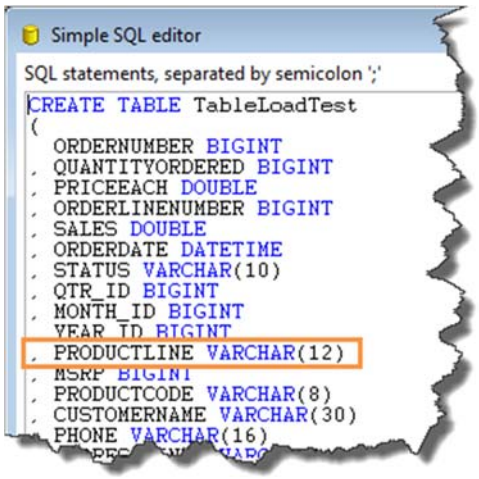
Create & Configure the Table Output Step

Step	Action
1	To create the next step, from the **Input** category of the **Design** tab, drag the **Table output** step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step** | **Destination Step** CSV file input | Table output
3	When creating the hop, you will be presented with a context menu. Click **Main output of step**. 
4	To open the step's properties dialog, double-click the step.
5	To configure the Filename property, in the **File** tab, set it according to the table shown below: **Property Name** | **Value** Connection | pentaho_olap Target table | SampleSizeTest 🕮 NOTE: All other properties on this tab should remain at their default values.
6	To generate the SQL query that will create the table defined in the Target table property, click the **[SQL]** button.

Continued on next page

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

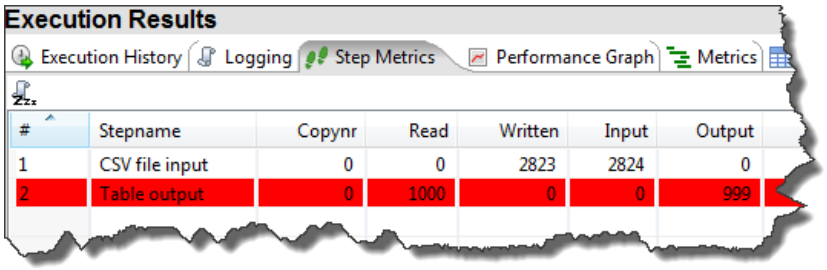
Create & Configure the Table Output Step, continued

Step	Action
7	Notice that the field lengths it is using to create the new table are based on those coming from the CSV file input step.  ■ Click **[Execute]** to create the new table. ■ Click **[OK]** to close the 'Results of the SQL statements' dialog. ■ Click **[Close]** to close the 'Simple SQL editor' dialog.
8	To close the step's properties dialog, click **[OK]**.
9	Save the transformation.

Continued on next page

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

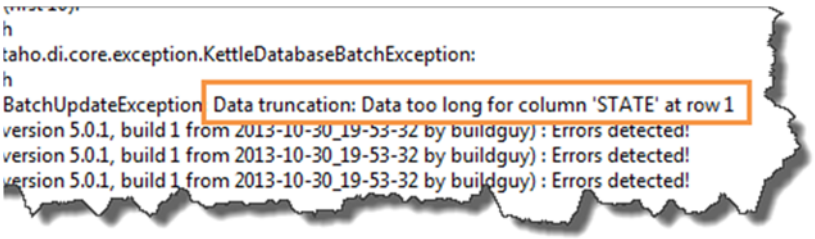
Execute the Transformation

Step	Action
1	To run the transformation, press **F9**, and then, click **[Launch]**.
2	Click the **Step Metrics** tab in the Execution Results pane to see that errors occurred. It should look similar to the screenshot shown below: 

Continued on next page

Guided Demo 9 – Choosing Adequate Sample Size for 'Get Fields', Continued

Execute the Transformation, continued

Step	Action
3	To see the exact errors causing the problem: ■ In the Execution Results pane, click the **Logging** tab. ■ Click the **Show error lines** icon. The error states that the data is too long for the column. In this case, the STATE column received the error first. 
4	To close the 'Error lines' dialog, click **[OK]**.

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Guided Demos`

Completed transformation:

`GD9_PreviewSampleSize.ktr`
(Use **File | Import from an XML file** to import)

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Exercise 8 – Lookups & Data Formatting

Introduction

This exercise introduces the concept of looking up data. The exercise scenario includes a flat file (.csv) of sales data that you will load into a database so that mailing lists can be generated. Several of the customer records are missing postal codes (zip codes) that must be resolved before loading into the database.

Objectives

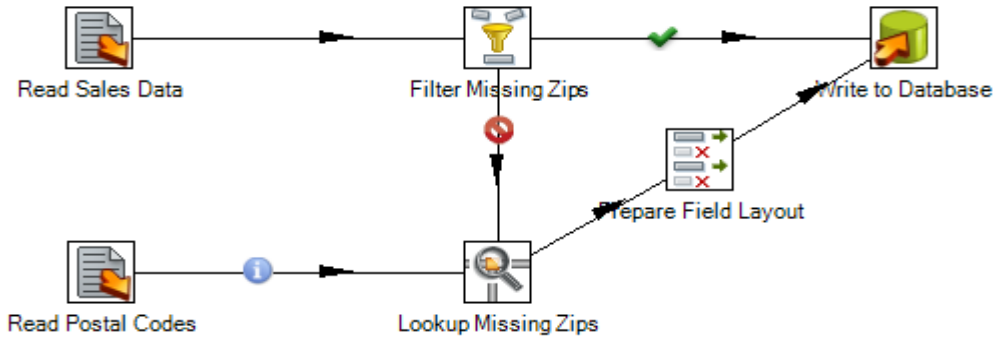
After completing this exercise, you will be able to:

- Merge Data from different streams.
- Use the Select Values step to change the name and format of a field.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Model



.

Exercise 8 – Lookups & Data Formatting, Continued

Using Stream Lookup with a Filter

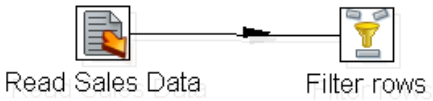
Step	Action
1	Open a new transformation
2	Save the transformation as **UpdateZip** in a location of your choice.
4	Add a **CSV file input** step. Then edit properties.
5	In the 'Step name' field, type **Read Sales Data**.
6	In the 'Filename' property, click **[Browse]**, and then locate the source file, **sales_data.csv**, found at: **C:\Pentaho\design-tools\data-integration\samples\transformations\files**
7	Make sure that the **Separator** is set to comma (,), and **Header** is enabled because there is one line of header rows in the file.

Continued on next page

Exercise 8 – Lookups & Data Formatting, Continued

Using Stream Lookup with a Filter

continued

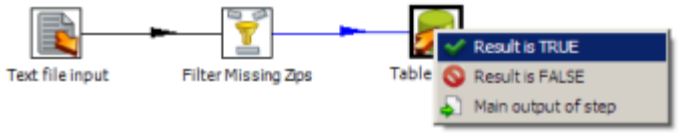
Step	Action
1	Click **[Get Fields]** to retrieve the input fields from your source file. A dialogue box will pop-up asking for a sample size (number of rows). Enter 0 (zero).
2	Click **Preview Rows** to verify that your file is being read correctly. You can change the number of rows to preview. Click **[OK]** to exit the step properties dialog box.
3	Add a 'Filter rows' step to your transformation. Under the **Design** tab, go to **Flow > Filter rows**. Drag it to the screen.
4	Create the hop, and make it the main path. 
5	**Double-click** the **Filter rows** step. The 'Filter Rows' properties dialog box appears.
6	In the **Step Name** field type, **Filter Missing Zips**.

Continued on next page

Exercise 8 – Lookups & Data Formatting, Continued

Using Stream Lookup with a Filter

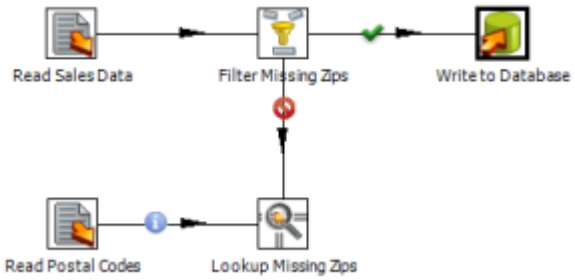
continued

Step	Action
7	In the **Fields:** dialog box select **POSTALCODE** and click **[OK]**
8	Click on the comparison operator (set to = by default) and select the **IS NOT NULL** function and click **[OK]**. Click **[OK]** to exit the 'Filter Rows properties' dialog box.
9	Click and drag a **Table Output** step into your transformation. Create a hop between the **Filter Missing Zips** (Filter Rows) and **Table Output** steps. In the dialog that appears, select **Result is TRUE.**  .
10	Double-click the **Table Output** step to open its edit properties dialog box.
11	Choose the **pentaho_olap** connection.
12	Type **SALES_DATA** in the 'Target table' field.
13	In the dialog box, **enable** the **Truncate table** property.
14	This table does not exist in the target database, so create it, perform the following: ■ Click the **[SQL]** button. ■ Click **[Execute]** to run the SQL. ■ Click **[OK]** to close the 'Simple SQL editor' dialog box. ■ Click **[OK]** to close the 'Table output' dialog box.
15	Save your transformation.
16	Add a new **CSV file input** step to your transformation. In this step you will retrieve the records from your lookup file.

Continued on next page

Exercise 8 – Lookups & Data Formatting, Continued

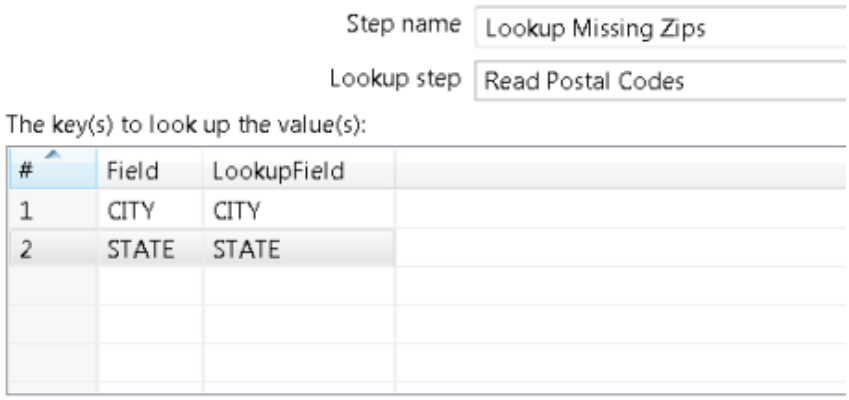
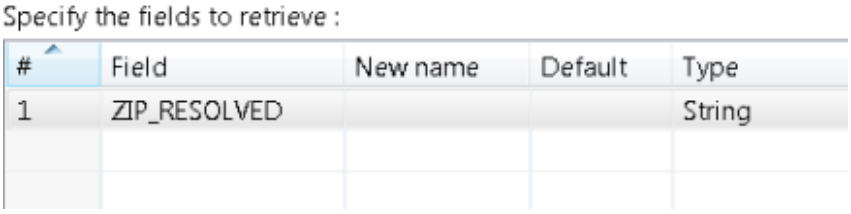
Using Stream Lookup with a Filter, continued

Step	Action
17	Rename the step to, **Read Postal Codes**.
18	Click **[Browse]** to locate the source file, **Zipssortedbycitystate.csv**, located at: **C:\Pentaho\design-tools\data-integration\samples\transformations\files**
19	Click **[Get Fields]** to retrieve the data from your .csv file. Rename the POSTALCODE field to **ZIP_RESOLVED**.
20	Click **[Preview Rows]** to make sure your entries are correct and click **[OK]** to close the step's properties dialog box.
21	Add a **Stream lookup** step to your transformation. Under the Design tab, expand the Lookup folder and choose Stream Lookup.
22	Draw a hop between the Filter Missing Zips and **Stream Lookup** steps. In the dialog box that appears, select Result is FALSE.
23	Create a hop from the **Read Postal Codes** step to the Stream lookup step. The transformation now looks like this. 
24	Double-click on the **Stream lookup** step to open its edit properties dialog box.
25	Rename Stream Lookup to **Lookup Missing Zips**

Continued on next page

Exercise 8 – Lookups & Data Formatting, Continued

Using Stream Lookup with a Filter, continued

Step	Action
26	From the Lookup step drop-down box, select **Read Postal Codes** as the lookup step.
27	Define the CITY and STATE fields in the key(s) to look up the value(s) table. Click the drop down in the Field column and select CITY. Then, click in the LookupField column and select CITY. Perform the same actions to define the second key based on the STATE fields coming in on the source and lookup streams: 
28	Click **[Get lookup fields]**. ZIP_RESOLVED is the only field you want to retrieve. To delete the CITY and STATE lines, right-click in the line and select Delete Selected Line and make sure the Type is set to **String**. Click **[OK]**. 

Continued on next page

Exercise 8 – Lookups & Data Formatting, Continued

Using Stream Lookup with a Filter

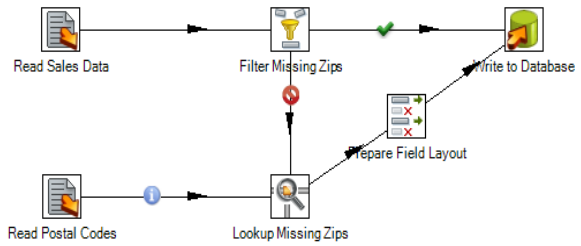
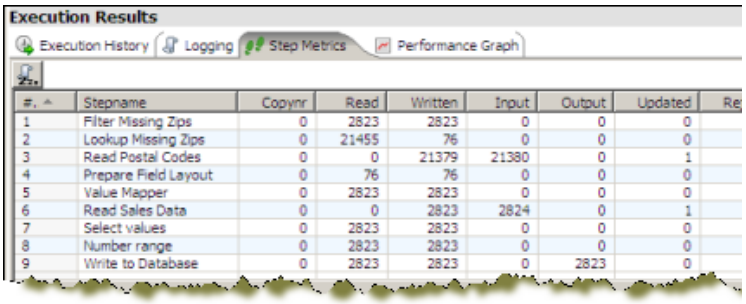
continued

Step	Action
29	In the canvas, select the **Lookup Missing Zips** step, right-click and select **Preview** to display the preview/debugger dialog box. Click **Quick Launch** to preview the data flowing through this step. Notice that the new field, ZIP_RESOLVED, has been added to the stream containing your resolved postal codes
30	Add a **Select values** step to your transformation by expanding the Transform folder and choosing Select Values.
31	Create a hop between the **Lookup Missing Zips** and **Select Values** steps.
32	Double-click the **Select Values** step to open its properties dialog box
33	**Rename** the 'Select value's step to **Prepare Field Layout**.
34	Click **[Get fields to select]** to retrieve all fields and begin modifying the stream layout.
35	Select the old **POSTALCODE** field in the list (line 20) and delete it.
36	Select the **ZIP_RESOLVED** field and press **CTRL-Up Arrow** until it is line #20.
37	The original POSTALCODE field was formatted as a 9-character string. You must modify your new field to match the form. Click the **Meta-Data** tab.
38	In the first row of the Fields to alter table, click in the Fieldname column and select ZIP_RESOLVED.
39	Configure this tab using the following steps: ■ Type **POSTALCODE** in the 'Rename to' column. ■ Select **String** in the 'Type' column. ■ Type **9** in the 'Length' column. ■ Click **[OK]** to exit the edit properties dialog box.

Continued on next page

Exercise 8 – Lookups & Data Formatting, Continued

Using Stream Lookup with a Filter, continued

Step	Action
40	Draw a **hop** from the **Prepare Field Layout** (Select values) step to the **Write to Database** (Table output) step, making it the main path. Save your transformation. 
41	In the canvas, select the Lookup Missing Zips step, right-click and select **Preview** to display the preview/debugger dialog box. Click **Quick Launch** to preview the data flowing through this step. Notice that the new field, ZIP_RESOLVED, has been added to the stream containing your resolved postal codes.
42	The transformation is complete. Run the Transformation.
43	Review the results. 

#.	Stepname	Copynr	Read	Written	Input	Output	Updated	Re
1	Filter Missing Zips	0	2823	2823	0	0	0	
2	Lookup Missing Zips	0	21455	76	0	0	0	
3	Read Postal Codes	0	0	21379	21380	0	1	
4	Prepare Field Layout	0	76	76	0	0	0	
5	Value Mapper	0	2823	2823	0	0	0	
6	Read Sales Data	0	0	2823	2824	0	1	
7	Select values	0	2823	2823	0	0	0	
8	Number range	0	2823	2823	0	0	0	
9	Write to Database	0	2823	2823	0	2823	0	

Solution Details

The solution to this exercise can be obtained using the details below:

Location: `C:\pentahotraining\Solutions\Exercises`

Completed transformation: `EX8_UpdateZip.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Guided Demo 10 – Creating Summary Fields Using Group By

Introduction

There are a variety of steps that can be used for calculations, this guided demo shows how to create summary data using **Group By** and the **Sort step.**

Objectives

After completing this guided demonstration, you will be able to:

- Create a new calculated field using 'Group by' step.
- Preview the results of the calculation
- Understand how the Sort step works.

Prerequisites

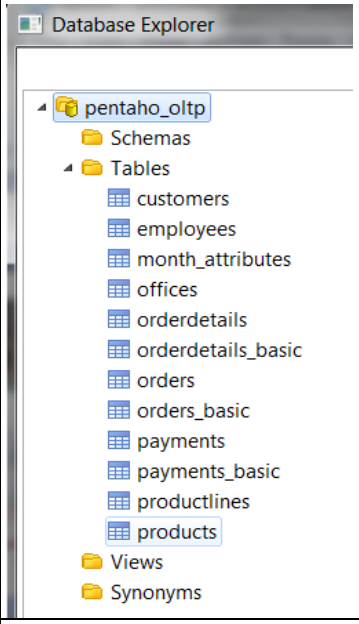
You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Model

Continued on next page

Guided Demo 10 – Creating Summary Fields Using Group By
Continued

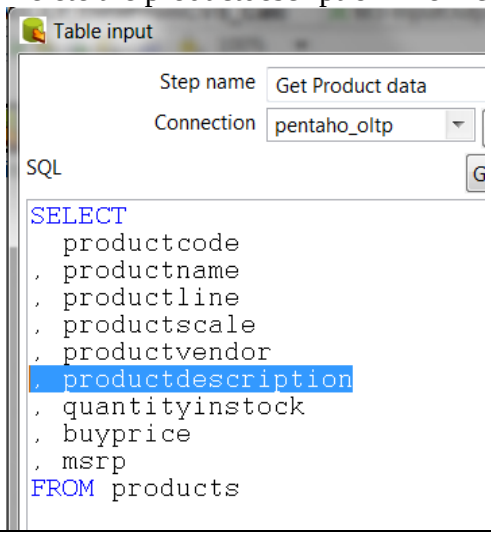
Creating the Transformation

Step	Action
1	Open a new transformation
2	Save the transformation as **SortGroupBy**.
3	From the Input Category drag the **Table input step** to the canvas.
4	Open the step by right clicking on it and choosing Edit Step from the dialogue.
5	Open the Text File Output step In the **Step Name** field type Get Product data.
6	In the connection drop-down choose pentaho_oltp
7	The database explorer will appear. Click on the key next to Tables, then click the key next to Tables this will show you a list of tables double click on the products table. 
8	A dialogue will appear asking if you'd like to include fieldnames in the SQL. Choose Yes.

Continued on next page

Guided Demo 10 – Creating Summary Fields Using Group By, Continued

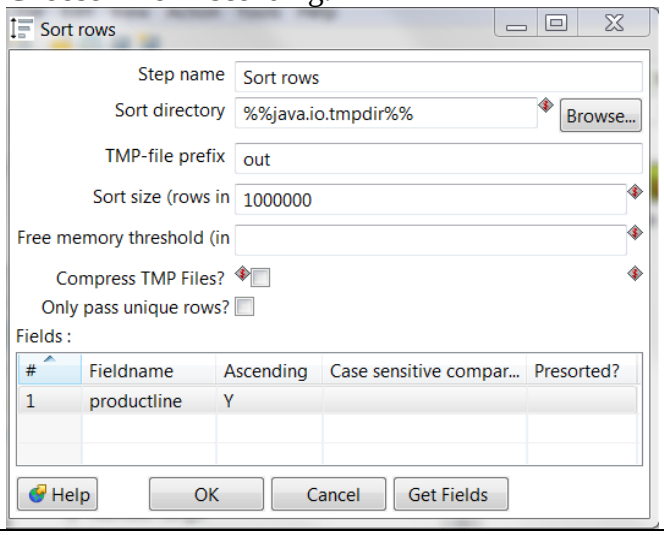
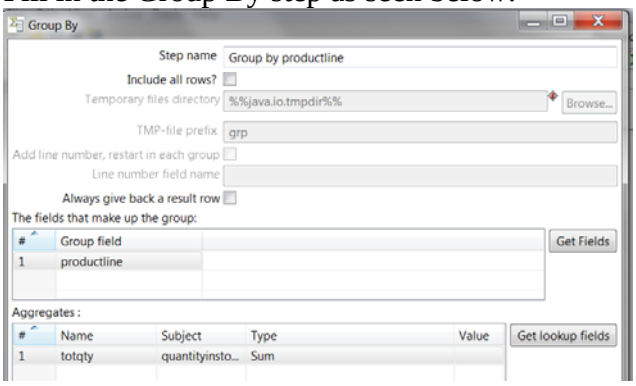
Creating the Transformation, continued

Step	Action
9	Delete the productdescription line including the leading comma. 
10	Click the **Preview** Find the field quantityinstock on the far right. That is the field we will be working with. Click **[OK]** to close the step.
11	From the Transform category drag the **Sort Rows** step to the canvas and place it to the right of the Table Input step.
12	Create the hop.
13	Open the Sort Rows step In the **Step Name** field type Sort by productline.

Continued on next page

Guided Demo 10 – Creating Summary Fields Using Group By, Continued

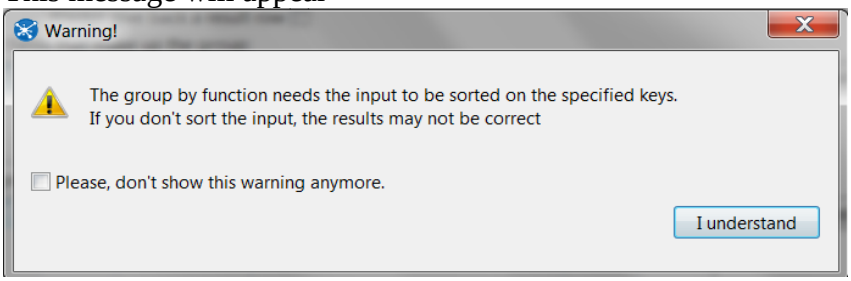
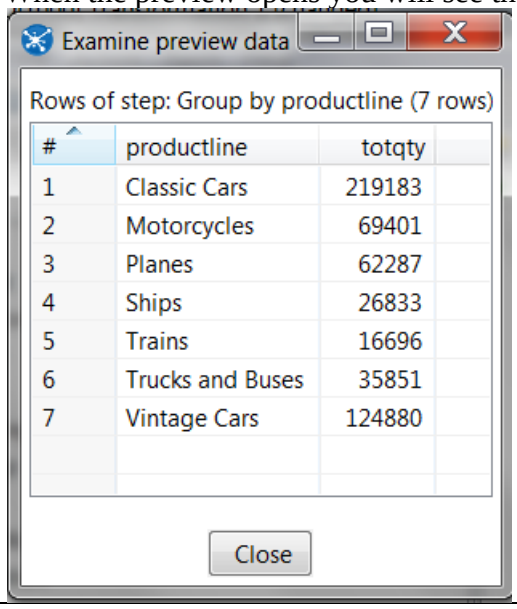
Creating the Transformation, continued

Step	Action
14	Open the Sort Rows step. Since we want to sort by productline, choose productline from the drop down list under Fieldname. Choose Y for Ascending. 
15	From the Statistics Category drag the **Group by** to the canvas.
16	Open the **Group by** step In the **Step Name** field type Group by productline. The transformation now looks like this. 
17	Fill in the Group By step as seen below. 

Continued on next page

Guided Demo 10 – Creating Summary Fields Using Group By, Continued

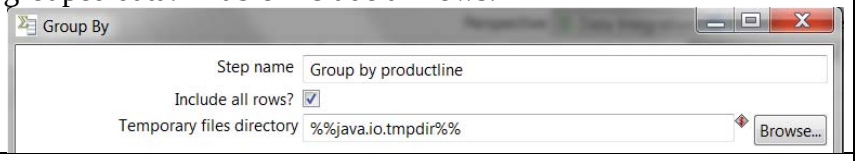
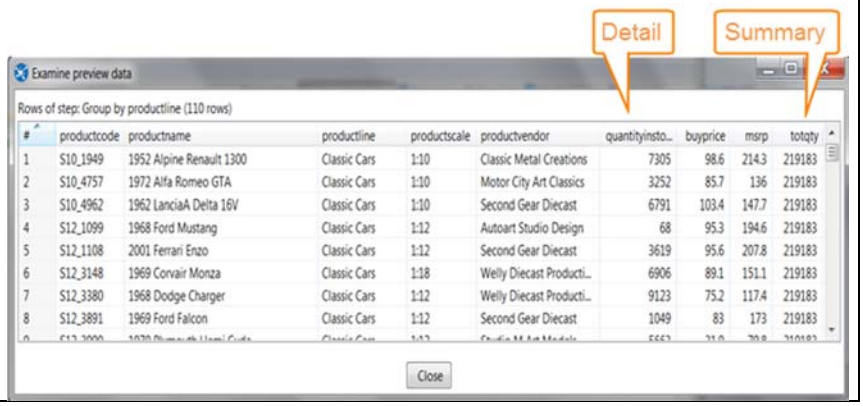
Creating the Transformation, continued

Step	Action
18	Click **[OK]** to close the step.
19	This message will appear 
20	The quick launch dialogue will appear clickG **[Quick Launch]**.
21	To see the results of the calculation right click on the step and choose preview.
22	The quick launch dialogue will appear click Quick Launch.
23	When the preview opens you will see the grouped data. 

Continued on next page

Guided Demo 10 – Creating Summary Fields Using Group By, Continued

Creating the Transformation, continued

Step	Action
24	Close the preview and reopen the **Group by** step. The include all rows option will show you both the individual records and the grouped data. Enable Include all rows. 
25	Close the step and run preview again. The result should look like this. 

#	productcode	productname	productline	productscale	productvendor	quantityinsto...	buyprice	msrp	totqty
1	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	7305	98.6	214.3	219183
2	S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	3252	85.7	136	219183
3	S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast	6791	103.4	147.7	219183
4	S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design	68	95.3	194.6	219183
5	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast	3619	95.6	207.8	219183
6	S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Producti...	6906	89.1	151.1	219183
7	S12_3380	1968 Dodge Charger	Classic Cars	1:12	Welly Diecast Producti...	9123	75.2	117.4	219183
8	S12_3891	1969 Ford Falcon	Classic Cars	1:12	Second Gear Diecast	1049	83	173	219183

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Guided Demos`

Completed transformation:
`GD10_SortGroupBy.ktr`
(Use **File | Import from an XML file** to import)

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Exercise 9 – Calculating & Aggregating Order Quantity

Introduction

In this exercise, you will create a transformation that reads the CSV file containing order data. Then, the data is sorted, by country and grouped by country for their total quantity ordered. Finally, calculates an individual order's quantity as a percentage of its country's total order quantity.

📖 NOTE

For those looking for more of a challenge, try the advanced version of this exercise. It is the same exercise, but without the detailed guidance. You will find it in this workbook immediately following this exercise.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Objectives

After completing this exercise, you will be able to:

- Sort rows of data on the stream.
- Group rows
- Use the Calculator step to calculate values based on values of data in the stream.

Continued on next page

Exercise 9 – Calculating & Aggregating Order Quantity, Continued

Create the Transformation

Step	Action
1	To create the new transformation, in the Spoon menubar, click **File | New | Transformation**.
2	To open the Transformation properties dialog, in the Canvas, double-click on an empty area.
3	Set the transformation properties for the **Transformation tab** according to the table below:

Property Name	Value
Transformation name	SortingData_Grouping
Description	Add a description of your choice.
Directory	/public/PDI_Trn_Objects 🕮 NOTE: This is in the repository.

Step	Action
4	To close the 'Transformation properties' dialog, click **[OK]**.
5	To save the transformation: ■ Press CTRL-S. ■ At the 'Transformation properties' dialog, click **[OK]**. ■ At the 'Enter a comment dialog', enter an optional comment, and then click **[OK]**. 🕮 NOTE: Now as you work on your transformation, you can easily save your work.

Continued on next page

Exercise 9 – Calculating & Aggregating Order Quantity, Continued

Create & Configure the CSV File Input Step

First, you will add a step to read all fields from a CSV file.

Step	Action
1	To create the first step, from the Input category of the Design tab, drag the CSV file input step onto the Canvas.
2	To open the step's properties dialog, double-click the step.
3	To configure the step's properties, set them according to the table shown below: **Property Name** / **Value** Step Name: Read Order Data From CSV File Filename: ${DIR_INPUT}\Order_File.csv Delimiter: ; Lazy conversion: (unchecked)
4	To configure the fields for this step: ■ Click the **[Get Fields]** button. ■ At the 'Sample size' dialog, click **[OK]**. ■ At the 'Scan results' dialog, click **[Close]**.
5	To close the step's properties dialog, click **[OK]**.
6	Save the transformation.

TIP

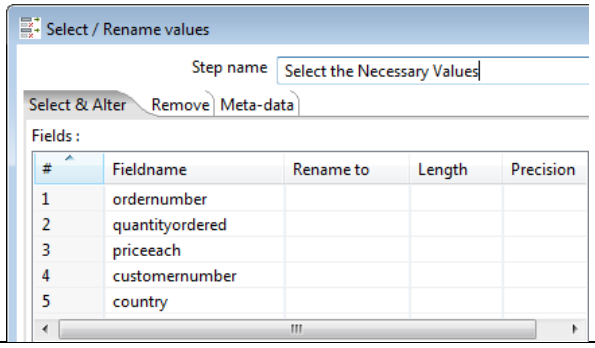
You can copy and paste a step from a previous transformation you created that has a CSV file input step configured the same way.

Continued on next page

Exercise 9 – Calculating & Aggregating Order Quantity, Continued

Create & Configure the Select Values Step

The first step reads all of the fields and data from the CSV file and adds them to the stream. Only a subset of those fields are needed, so you will add a step that will only select the fields that contain the data you are interested in.

Step	Action
1	To create the second step, from the Transform category of the Design tab, drag the Select /Rename values step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step**: CSV file input **Destination Step**: Select / Rename values
3	To open the step's properties dialog, double-click the step.
4	To configure the step's properties, set them according to the table shown below: **Property Name**: Step Name → **Value**: Select the Necessary Values **Property Name**: Select & Alter Tab: Fieldname → **Value**: ordernumber, quantityordered, priceeach, customernumber, country
5	Compare the step's configuration with the screenshot below and make any necessary changes. 
6	To close the step's properties dialog, click **[OK]**.
7	Save the transformation.

Continued on next page

Exercise 9 – Calculating & Aggregating Order Quantity, Continued

Create & Configure the Sort Rows Step

The next step you will add will sort the data stream by country.

Step	Action
1	To create the next step, from the **Transform** category of the **Design** tab, drag the **Sort rows** step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step** → **Destination Step** Select / Rename values → Sort rows
3	To open the step's properties dialog, double-click the step.
4	To configure the step's properties, set them according to the table shown below: **Property Name** / **Value** Step Name: Sort Rows by Country
5	To configure the field to sort by, in the Fields grid, add a row according to the table below: **Fieldname** / **Ascending** / **Case sensitive compare** / **Presorted** country / Y / N / N
6	To close the step's properties dialog, click **[OK]**.
7	Save the transformation.

Continued on next page

Exercise 9 – Calculating & Aggregating Order Quantity, Continued

Create & Configure the Group by Step

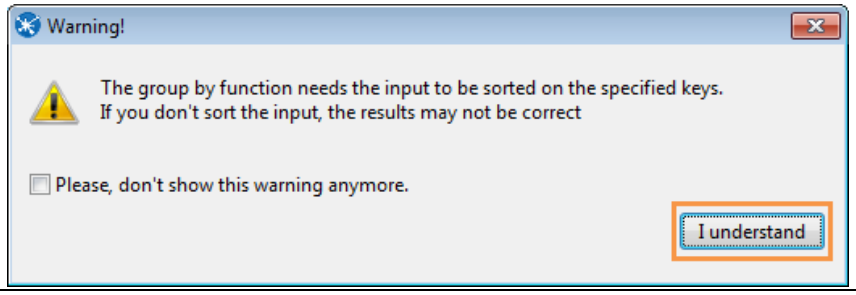
The next step you will add will group the data by country and calculate the total quantity ordered for each country. The values will be added to each row in a new field called TotalQtyOrdered.

Step	Action
1	To create the next step, from the **Statistics** category of the **Design** tab, drag the **Group by** step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step** | **Destination Step** Sort rows | Group by
3	To open the step's properties dialog, double-click the step.
4	To configure the step's properties, set them according to the table shown below: **Property Name** | **Value** Step Name | Aggregate Quantity Ordered Include all rows? | (checked)
5	To configure the field to group by, in **The fields that make up the group** grid, add a row according to the table below: **#** | **Group field** 1 | country
6	Configure the Aggregates grid, by adding a row according to the table below: **Name** | **Subject** | **Type** TotalQtyOrdered | quantityordered | Sum
7	To close the step's properties dialog, click **[OK]**.

Continued on next page

Exercise 9 – Calculating & Aggregating Order Quantity, Continued

Create & Configure the Group by Step, continued

Step	Action
8	To confirm that you understand data must be sorted when using this step, click **[I understand]**. 
9	Save the transformation.

Create & Configure the Calculator Step

The final step in this transformation is used to calculate for each group (country), the percentage of the total quantity ordered that was ordered in each row.

Step	Action
1	To create the next step, from the **Transform** category of the **Design** tab, drag the **Calculator** step onto the Canvas.
2	Create a hop between the steps as shown in the table below: **Source Step** | **Destination Step** Group by | Calculator
3	To open the step's properties dialog, double-click the step.
4	To configure the step's properties, set them according to the table shown below: **Property Name** | **Value** Step Name | Calculate Percentage

Continued on next page

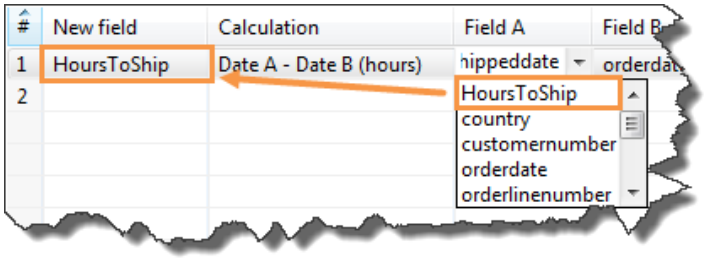
Exercise 9 – Calculating & Aggregating Order Quantity, Continued

Create & Configure the Calculator Step, continued

Step	Action
5	To configure the **Fields** grid, add a row according to the table below: **New field**: %_Ordered ForCountry **Calculation**: 100 * A / B **Field A**: quantityordered **Field B**: TotalQtyOrderd
6	To close the step's properties dialog, click **[OK]**.
7	Save the transformation.

📖 **NOTE**

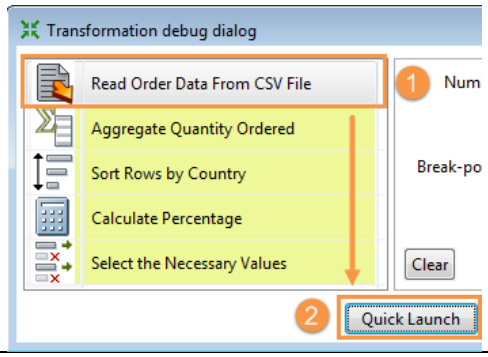
When configuring the calculator step's field grid, Spoon automatically adds each new field you define to the list of available fields for subsequent rows. This terrific feature allows you to use the field you just defined in other calculations in the same step.



Preview the Steps

This transformation does not create an output file or insert data into a database, therefore you must preview the last step to verify it functions properly. Before doing this however, it is recommended that you preview each step, starting with the first, to make sure the stream contains the data you expect.

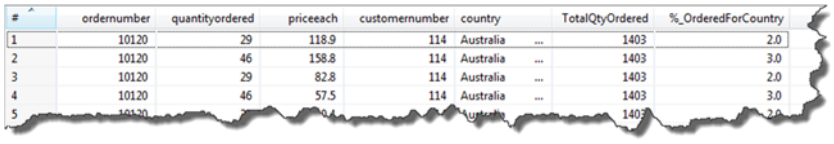
Step	Action
1	To select all of the steps, in the Canvas, use the mouse to drag a selection box around all of the steps.
2	To preview the transformation: ■ Click the Preview button in the Canvas toolbar. ■ Select the step to preview in the 'Transformation debug' dialog.

	■ Click **[Quick Launch]**. 

Continued on next page

Exercise 9 – Calculating & Aggregating Order Quantity, Continued

Preview the Steps, continued

Step	Action
3	Verify the data in the preview is what was expected, and then, click **[Close]**.
4	Preview the remaining steps. The last step's preview should look like the example screenshot below: 

#	ordernumber	quantityordered	priceeach	customernumber	country	...	TotalQtyOrdered	%_OrderedForCountry
1	10120	29	118.9	114	Australia	...	1403	2.0
2	10120	46	158.8	114	Australia	...	1403	3.0
3	10120	29	82.8	114	Australia	...	1403	2.0
4	10120	46	57.5	114	Australia	...	1403	3.0
5	10120							

Solution Details

The solution to this exercise can be obtained using the details below:

Location:
`C:\pentahotraining\Solutions\Exercises`

Completed transformation:
`EX9_SortingData_GroupingSelectValues.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulations! You have completed this exercise.

Exercise 9 Advanced – Calculating & Aggregating Order Quantity

Introduction

This is an advanced version of Exercise 9. It is the same exercise, but without the detailed guidance. You may choose to do this advanced exercise rather than the standard version of Exercise 9.

In this exercise, you will create a transformation that reads the CSV file containing order data. Then, the data is sorted, by country and grouped by country for their total quantity ordered. Finally, the percentage of the quantity ordered is calculated.

Instructions

Create a transformation that has the following functionality:

- Reads in the file: C:\pentahotraining\DataFiles\Order_File.csv
- Uses only the following fields from the file:
  - order_number
  - quantityordered
  - priceeach
  - customernumber
  - country
- Sorts the data from by country.
- Determines the total quantity of items ordered per country.
- Calculates an individual order's quantity as a percentage of its country's total order quantity.
- Uses parameters for any file input and output locations.

Steps Used

Use only the steps shown below to complete this exercise:

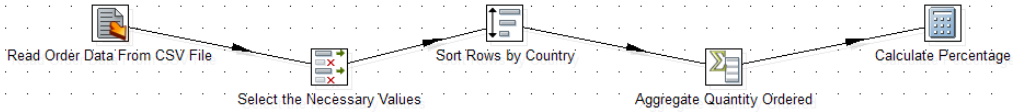
- CSV file input
- Select values
- Sort rows
- Group by
- Calculator

Continued on next page

Exercise 9 Advanced – Calculating & Aggregating Order Quantity, Continued

Finished Transformation

The completed transformation should resemble the screenshot shown below:



Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

End of Exercise

Congratulations! You have completed this exercise.

Exercise 10 – Loading JVM Data into a Table

Introduction

Monitoring memory is an important part of making sure applications written in Java, like PDI, execute efficiently. In this exercise, you create a job that runs an existing transformation that reads the Java Virtual Memory (JVM) heap data and loads it into a database table. You define the database table's name as a PDI environmental variable.

📖 NOTE

For those looking for more of a challenge, try the advanced version of this exercise. It is the same exercise, but without the detailed guidance. You will find it in this workbook immediately following this exercise.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Objectives

After completing this exercise, you will be able to:

- Open and understand one of the sample transformations that come with PDI.
- Set PDI environmental variables.
- Create a new job and configure it to execute a transformation.
- Execute a job.

Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

Open & Explore the JVM Health Check Transformation

In this portion of the exercise, you will open an existing transformation that collects JVM data and loads it into a database. Then, you will explore its operation.

Step	Action
1	To open the existing transformation: ■ In the Spoon menubar, click **File | Import from an XML file**. ■ Navigate to: **C:\pentahotraining\Samples and Demos\Operational Patterns\JVM_health_check** ■ **Double-click** the transformation named: **JVM_collect_data.ktr**
2	Spend a few minutes to explore each step's properties. You should be able to determine that the transformation performs these basics functions in its steps: ■ Retrieves various JVM related data. ■ Calculates available JVM memory and percentage. ■ Retrieves the value of the variable named operations_instance_id. ■ Selects only the fields from the stream that are wanted. ■ Loads the selected field values into a table. The table's name is in a variable.
3	To save the transformation, in the Spoon menubar, click File | Save As, and then, set the properties as shown in the table below:
4	To close the 'Job properties' dialog, click **[OK]**.

Property Name	Value
Transformation name	EX10_JvmCollectData
Description	Add a description of your choice.
Directory	/public/PDI_Trn_Objects 🕮 NOTE: This is in the repository.

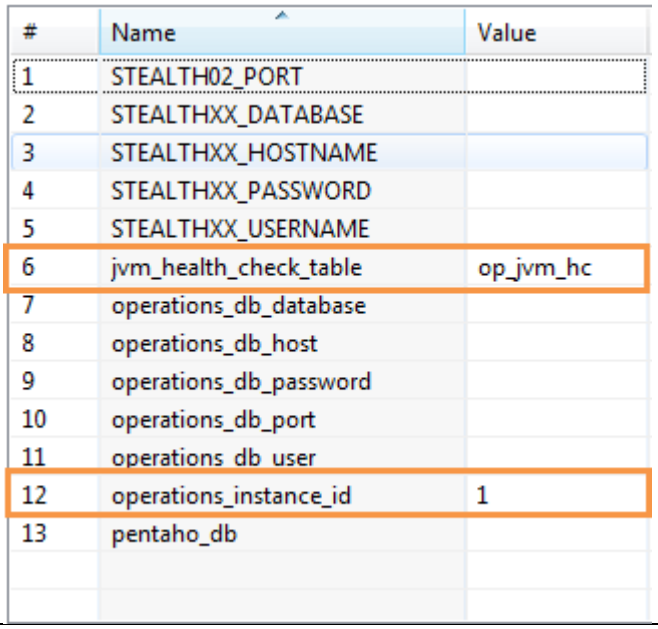
Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

Set Environmental Variables

You have learned at a high level, how this transformation functions, and noticed how it uses variables. In this section, you will set the values of those variables.

🕮 NOTE: The transformation note that contains the CREATE TABLE statement shows that the table's name should be 'op_jvm_hc'. The operations_instance_id that is retrieved by the get variables step is provided as an example and its value can be any integer.

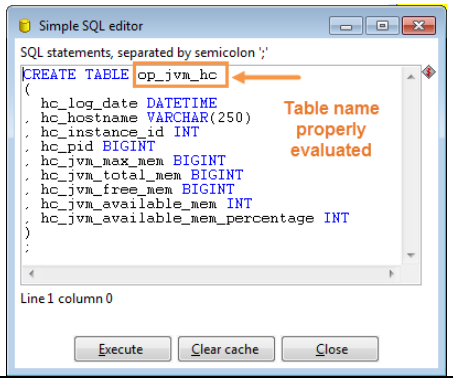
Step	Action
1	To open the 'PDI environmental variables' dialog, in the Spoon menubar, click **Edit | Set Environment Variables**.
2	To set the two variables, set the value of the Value column as shown in the table below: **Name** / **Value** jvm_health_check_table / op_jvm_hc operations_instance_id / 1 
3	To close the 'Set Environment Variables' dialog, click **[OK]**.

Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

Set the Database Connection & Create Table

The last step in this transformation loads the JVM data into a table. You must define which database connection to use, and then create the table into which the JVM data will be stored.

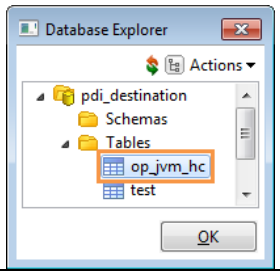
Step	Action
1	To set the database connection to use, **double-click** the **${jvm_health_check_table}** step, and then set its properties according to the table below. **Property Name** / **Value** Connection / pentaho_olap IMPORTANT: Notice the Target table property is using a variable. This is one of the variables you set a value to in the previous section.
2	To open the Simple SQL editor, click the **[SQL]** button. Notice how PDI automatically created a create table statement and evaluated the correct name of the table to create based on the value you set the environmental variable to. 
3	To create the table: ■ Click the **[Execute]** button. ■ Click **[OK]** to close the 'Results of the SQL statements' dialog. ■ Click **[OK]** to close the 'Simple SQL Editor' dialog.
4	To close the 'Table output' step's dialog, click **[OK]**.

Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

Verify Table Creation

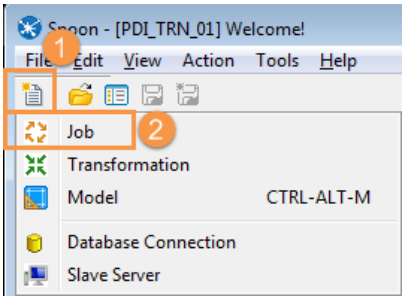
TIP: It is best practice to confirm the table was created successfully. Spending time to verify this now may help prevent a lengthy troubleshooting session. If you were to try loading data into a table that does not exist, errors will result.

Step	Action
1	To open the Database Explorer: ■ In the Spoon menubar, click Tools | Database | Explorer. ■ Click 'pentaho_olap'. ■ Click **[OK]**.
2	To verify the table was created: ■ In the Database Explorer, expand **pentaho_olap**. ■ Expand **Tables**. ■ Notice the existence of the **op_jvm_hc** table as shown in the screenshot below. 
3	To close the Database Explorer, click **[OK]**.

Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

Create the Job

Step	Action
1	To create the new job, in the Spoon **toolbar**, click the **New file** icon, and then click **Job**.  Shortcut: You can also use CTRL-ALT-N to create a new job.
2	To open the 'Job properties' dialog, in the Canvas, double-click on an empty area.
3	Set the transformation properties for the **Job tab** according to the table below: **Property Name** / **Value** Job name: LoadJvmDataToTable Description: Add a description of your choice. Directory: /public/PDI_Trn_Objects 🕮 NOTE: This is in the repository.
4	To close the Job dialog, click **[OK]**.
5	To save the job: ■ Press **CTRL-S**. ■ At the 'Job' dialog, click **[OK]**. ■ At the 'Enter a comment' dialog, enter an optional comment, and then click **[OK]**. 🕮 NOTE: Now as you work on your job, you can easily save your work.

Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

Create & Configure the Steps

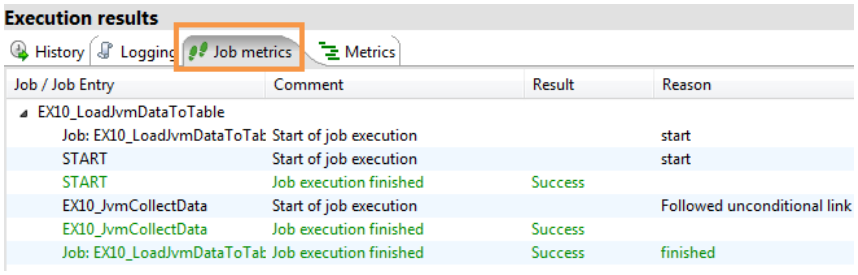
Step	Action
1	To create the start step, from the **General** category of the **Design** tab, drag the **Start** step onto the Canvas.
2	To create the step that runs the transformation, from the **General** category of the **Design** tab, drag the **Transformation** step onto the Canvas.
3	Create a hop between the steps as shown in the table below: **Source Step** | **Destination Step** Start | Transformation
4	To open the Transformation step's properties dialog, double-click the Transformation step.
5	To set the step's properties, set them according to the table shown below: **Property Name** | **Value** Name of job entry | Run JVM Data to Table
6	To configure the name and location of the transformation this step will execute, in the **Transformation specification** tab, select the **Specify by name and directory** radio button, and then set its values as shown below using the Transformation selector button :  📖 NOTE: If you saved your transformation in the file system, use the 'Transformation filename' radio button and property
7	To close the job's properties dialog, click **[OK]**.
8	Save the transformation.

Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

Execute the Job & Verify Table Load

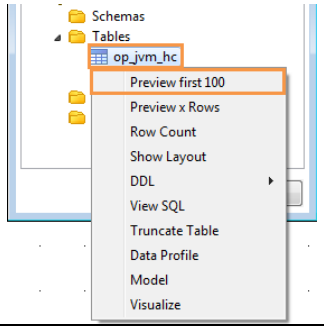
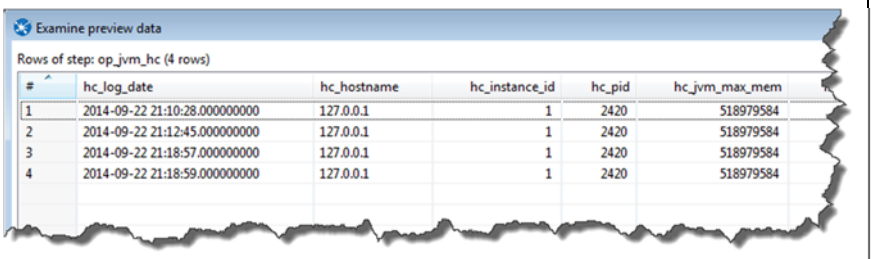
In the final section of this exercise, you will execute the job, verify it can successfully, and verify the table was loaded with data.

Step	Action
1	To run the job, press **F9**, and then click **[Launch]**.
2	There should be no errors and the **Job Metrics** tab should look similar to the screenshot shown below: 
3	To have several rows of data loaded into the output table, **run** the **job** a **few more times**.
4	To open the Database Explorer: ■ In the Spoon menubar, click **Tools | Database | Explorer** ■ Click 'pentaho_olap'. ■ Click **[OK]**.

Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

Execute the Job & Verify Table Load, continued

Step	Action
5	To verify the table was loaded: ■ In the Database Explorer, expand **pentaho_olap**. ■ Expand **Tables**. ■ Right-click the **op_jvm_hc** table. ■ Click **Preview first 100**. 
6	Verify that you have several rows of data loaded in the table as shown in the sample screenshot below: 
7	To close the 'Examine preview data' dialog, click **[OK]**.
8	To close the Database Explorer, click **[OK]**.

Continued on next page

Exercise 10 – Loading JVM Data into a Table, Continued

TIP

You can set the JAVA HEAP size for Spoon by editing the **spoon.bat** or **spoon.sh** files. The files are located in the following folder (default location):

`C:\Pentaho\design-tools\data-integration`

By default, the maximum heap size is only set to 512 MB. Most uses for PDI require setting a larger maximum heap size.

Solution Details

The solution to this exercise can be obtained using the details below:

Location:

`C:\pentahotraining\Solutions\Exercises\`

Completed job: `EX10_LoadJvmDataToTable.kjb`
Completed transformation: `EX10_JvmCollectData.ktr`
(Use **File | Import from an XML file** to import)

End of Exercise

Congratulation! You have completed this exercise.

Exercise 10 Advanced – Loading JVM Data into a Table

Introduction

This is an advanced version of Exercise 10. It is the same exercise, but without the detailed guidance, and the addition of using a parameter. You may choose to do this advanced exercise rather than the standard version of Exercise 10.

Monitoring memory is an important part of making sure applications written in Java, like PDI, execute efficiently. In this exercise, you create a job that runs an existing transformation that reads the Java Virtual Memory (JVM) heap data and loads it into a database table. You define the database table's name as a PDI environmental variable.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Instructions & Objectives

This exercise contains several parts. Please read each part section below to see the requirements and instructions necessary for completing each part this advanced exercise.

After completing all parts of this exercise, you will be able to:

- Open and understand one of the sample transformations that come with PDI.
- Set PDI environmental variables.
- Create a new job and configure it to execute a transformation.
- Execute a job.

Part I

Open the transformation shown in the table below and examine its steps and their properties. Spend some time learning what this transformation does and how it works.

Field	Value
Transformation Location	C:\pentahotraining\Samples and Demos\Operational Patterns\JVM_health_check
Transformation Name	JVM_collect_data.ktr

Continued on next page

Exercise 10 Advanced – Loading JVM Data into a Table, Continued

Part II

In this part of the exercise, you modify the transformation and set environmental variable values.

Step	Action
1	Set the environmental variables to the values shown in the table below. **Environment Variable Name** / **Value** jvm_health_check_table / op_jvm_hc operations_instance_id / 1
2	Modify the transformation so that it is loading the data into the 'pentaho_olap' database. Use the environment variable to define which table in the database should be loaded.

Part III

Create a job that has the following functionality:

- Executes the transformation from Part II of this exercise.

Part IV

Verify the following:

- The transformation executed successfully.
- The expected table was loaded with the JVM data.

End of Exercise

Congratulations! You have completed this exercise.

Exercise 11 – Using the Pentaho Enterprise Repository

Introduction

Up to this point in the course you have used the PDI Repository only to save your transformations, and perhaps your jobs as well. The PDI Repository has features that go far beyond saving your objects. In this exercise, you get acquainted with those features. The exercise is broken into two parts, I & II.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Objectives

After completing this exercise, you will be able to:

- Create a new connection to the Pentaho Enterprise Repository.
- Configure the Pentaho Enterprise Repository, including basic security options.
- Create folders in the repository.
- Add transformations and jobs to the repository.
- Move, lock, and revise artifacts stored in the repository.
- Delete and restore repository artifacts.

Part I: Using the enterprise repository

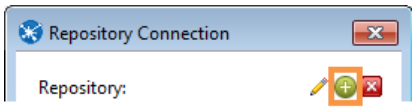
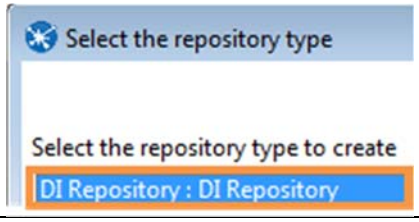
In part I of this exercise you create a new connection to the Pentaho Enterprise Repository, and then perform common tasks such as creating folders in the repository, adding transformations and jobs to the repository, locking repository artifacts, revising repository artifacts, and setting basic security options.

Continued on next page

Exercise 11 – Using the Pentaho Enterprise Repository, Continued

Creating a Connection to the Repository

To create a connection to the enterprise repository:

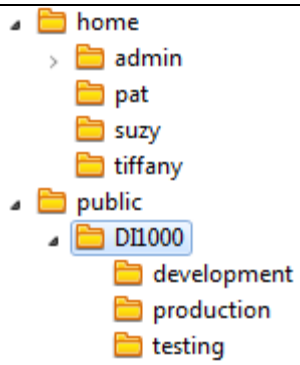
Step	Action
1	Close Spoon if is open, saving any of your previous work first.
2	If the Data Integration server is not running, start it by choosing **Start > Programs > Pentaho Enterprise Edition > Server Management > Start Data Integration Server**. 🕮 NOTE: For standalone PDI installs, the DI Server may not start without Java. Your instructor can provide the Java installer.
3	Launch Spoon.
4	To open the Repository Connection dialog, in the menubar, click **Tools | Repository | Connect**. ☝ TIP: You can use the keyboard shortcut **CTRL-R** to quickly open the Repository Connection dialog.
5	At the 'Repository Connection' dialog, click the **Add Connection** icon (the green + icon). 
6	When prompted to choose the repository type, select the first one, **DI Repository**. 
7	Click **[OK]**.

8	At the 'Repository Configuration' dialog, enter the following URL exactly as shown. The Name and Description may be anything you like, as long as the Name is unique among other connections. **Field** / **Value** URL: http://localhost:9080/pentaho-di ID: PDI_Class Name: This connection is created in class to demonstrate working with the respository and its artifacts.
9	Click **[OK]**.
10	When you are returned to the 'Repository Connection' dialog, select your repository and enter the following: **Field** / **Value** User Name: admin Password: password 🕮 NOTE: If the repository connection fails, verify the Data Integration server is running.
11	Click **[OK]**.

Explore the repository structure

To explore the structure of the repository:

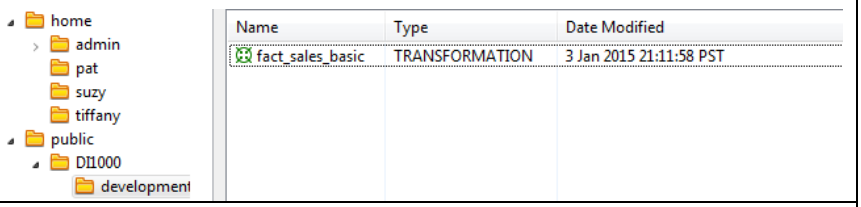
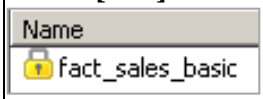
Step	Action
1	Click the **Explore Repository** icon.  TIP: You can use the keyboard shortcut **CTRL-E** to quickly open the Repository Explorer dialog.
2	On the **Browse** tab, you should see a single private folder in the `\home` folder labeled with your name: **admin**.
3	Right-click **Public** and choose **New Folder**.
4	Name the folder **DI1000** and click **[OK]**.
5	Using the previous steps, create 3 new folders under DI1000 named: **development**, **testing**, and **production**.

	
6	Close the Repository Explorer.
7	From the menu options, choose **File | Import from an XML file**.

Continued on next page

Exercise 11 – Using the Pentaho Enterprise Repository, Continued

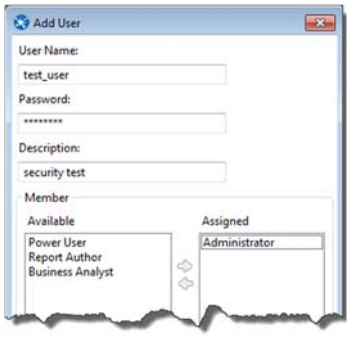
Explore the repository structure (continued)

Step	Action
8	Navigate to the folder shown below and select **fact_sales_basic.ktr**, and then click **[OK]**. **C:\pentahotraining\Support Files\EX11_Repository**
9	Click **File** | **Save as**.
10	In the 'Transformation properties' dialog, for **Directory**, click the "Select a folder…" button to select a repository folder.
11	Select **`\public\DI1000\development`** and click **[OK]**.
12	Click **[OK]** to close the "Transformation properties."
13	When the 'Enter comment' dialog opens, type a comment of your choice.
14	Click **[OK]**.
15	Click the **Explore Repository** icon.
16	Expand **public\DI1000** and click **development**.
17	Notice that the file name, type and modification date appear in the document window. 
18	Drag **fact_sales_basic** from the **development** folder into **`\home\admin`**.
19	Click **`\home\admin`** to verify the file was transferred successfully.
20	Right-click **fact_sales_basic** and choose **Lock**.
21	In the 'Lock Notes' dialog, type: **Locked by Admin for testing**.
22	Click **[OK]**. Notice the file icon changes to a padlock. 
23	Click the **Security** tab.
24	Click the **Add User** button (green +).

Continued on next page

Exercise 11 – Using the Pentaho Enterprise Repository, Continued

Explore the repository structure (continued)

Step	Action
25	Type the following in the' Add User' dialog: **Field** / **Value** User Name: `test_user` Password: `password` Description: `security test`
26	In the **Member** section, in the **Available** window, select **Admin** and click the right-arrow to add the role to the **Assigned** window.  🕮 NOTE: You may need to expand the size of the dialog to see the Member section.
27	Click **[OK]**.
28	Close the Repository Explorer.
29	Close **fact_sales_basic.ktr**.
30	From the menu, choose **Tools | Repository | Disconnect repository**.
31	Choose **Tools | Repository | Connect**.
32	In the 'Repository Connection' dialog, choose your repository and log in using: **User Name**: `test_user` **Password**: `password`
33	Open the **Repository Explorer**.
34	Notice you can view all user folders and public folders.

Continued on next page

Exercise 11 – Using the Pentaho Enterprise Repository, Continued

Explore the repository structure (continued)

Step	Action
35	Click the **Security** tab and select **test_user**.
36	Click the **Edit** icon (pencil).
37	In the **Member** section, remove the **Admin** role and click **[OK]**.
38	Close the Repository Explorer.
39	Using previous steps, disconnect from the repository and re-connect as **test_user**.
40	Open the **Repository Explorer** and notice you can only view public folders (plus your own user folder).
41	Close the Repository Explorer and disconnect from the repository.

Part II: Repository artifacts

In part II of this exercise you work with the repository artifacts you configured previously, including: restoring deleted artifacts and working with locked artifacts.

Working with repository artifacts

To work with repository artifacts:

Step	Action
1	Connect to the repository using: User Name: **`admin`** Password: **`password`**
2	Open the **Repository Explorer**.
3	Unlock the transformation and Drag the unlocked transformation, **fact_sales_basic**, from **`\home\admin`** into the **`public\DI1000\development`** folder.
4	Close the Repository Explorer, and disconnect from the repository.
5	Connect to the repository using the **test_user** credentials.
6	Open the **Repository Explorer**, expand **`public\DI1000`**, and click **development**.
7	Double-click **fact_sales_basic.ktr** (to open it) and close the Repository Explorer.

Continued on next page

Exercise 11 – Using the Pentaho Enterprise Repository, Continued

Working with repository artifacts (continued)

Step	Action
8	Right-click one of the notes and choose **Delete note** from the menu.
9	Notice that the Save icon is dimmed. Though you are allowed to make a change to the locked file, you cannot save it. You may only use the "Save as" option to create a copy.
10	Close fact_sales_basic and do not save the file.
11	Disconnect from the repository.
12	Re-connect using **admin's** credentials.
13	Open the **Repository Explorer** and open **`public\DI1000\development\fact_sales_basic.ktr`**.
14	Close the Repository Explorer.
15	Delete a note from the transformation, save it, and close it.
16	When the 'Enter comment 'dialog opens, type: **fact_sales_basic V2** and click **[OK]**.
17	Open the **Repository Explorer**.
18	Expand **`public\DI1000\`** and click **`development`**.
19	Click **fact_sales_basic** and view the **Version History**. Version History | Access Control Label / Date Created / Comment / User 1.0 — 12 Aug 2012 18:03:05 EDT — fact_sales_basic imported into dev repo — joe 1.1 — 12 Aug 2012 20:33:28 EDT — fact_sales_basic V2 — joe
20	Close the Repository Explorer.

End of Exercise Congratulations! You have completed this exercise.

Guided Demo 11 – Scheduling & Monitoring

Introduction

In this guided demo you will configure job scheduling and monitoring using PDI.

Objectives

After completing this guided demo, you will be able to:

- Schedule the execution of a transformation using Pentaho Data Integration
- Monitor job execution on the Pentaho Data Integration server
- View transformation step metrics in Carte
- View performance trend data in Carte

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Scheduling a Transformation

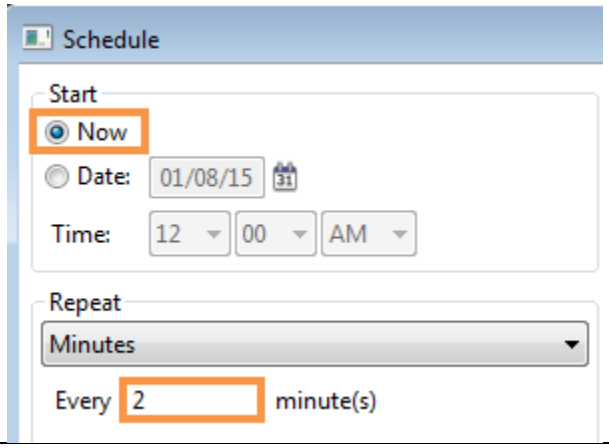
To schedule a transformation:

Step	Action
1	If the Data Integration server is not running, start it by choosing **Start > Programs > Pentaho Enterprise Edition > Server Management > Start Data Integration Server**. 🕮 NOTE**:** Type localhost:9080 into a browser. If you see the login screen your server is running.
2	In PDI, choose **Tools | Repository | Connect** from the menu.

Continued on next page

Guided Demo 11 – Scheduling & Monitoring, Continued

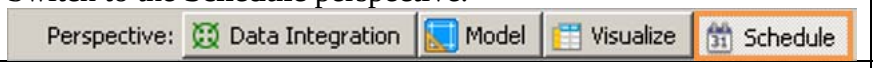
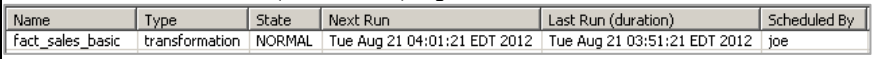
Scheduling a Transformation, continued

Step	Action
3	In the 'Repository Connection' dialog, choose your repository and log in using: User Name: **`admin`** Password: **`password`**
4	Click the **Explore Repository** icon.
5	Expand **public** > **DI1000** and click **development**.
6	Open the **fact_sales_basic** transformation.
7	Close the Repository Explorer.
8	From the menu options, choose **Action** | **Schedule**.
9	In the 'Schedule' dialog set the following options: **Option** / **Setting** Start: **`Now`** Repeat: **`Minutes`** Every minute(s): **`2`** 

Continued on next page

Guided Demo 11 – Scheduling & Monitoring, Continued

Scheduling a Transformation, continued

Step	Action
10	Notice the other options for scheduling transformations and jobs and click **[OK]** when you are finished.
11	Switch to the **Schedule** perspective. 
12	Examine the scheduling options. In particular, notice the **Next Run** and **Last Run (duration)** options:  When you are finished, switch to the **Data Integration** perspective.

Name	Type	State	Next Run	Last Run (duration)	Scheduled By
fact_sales_basic	transformation	NORMAL	Tue Aug 21 04:01:21 EDT 2012	Tue Aug 21 03:51:21 EDT 2012	joe

Monitoring a Slave Server

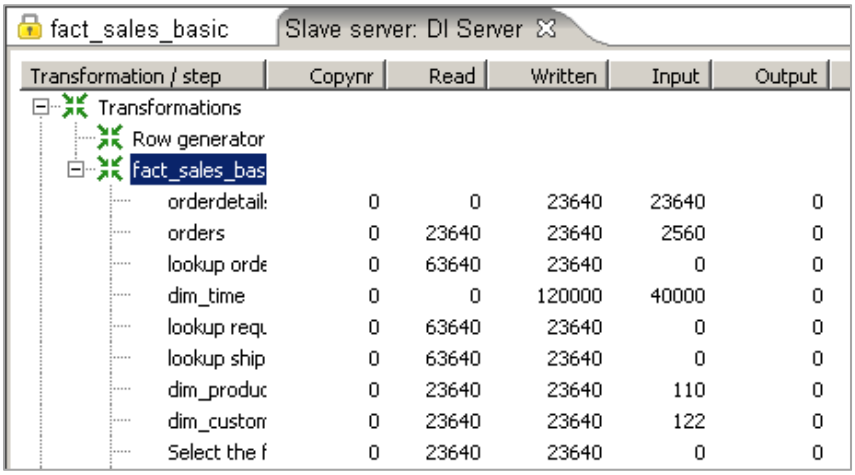
To monitor transformation execution on a slave server:

Step	Action
1	On the **View** tab, expand **Transformations** > **fact_sales_basic** (if necessary).
2	Right-click **Slave server** and choose **New**.
3	In the 'Slave server dialog' dialog, on the **Service** tab, type the following: **Field** / **Value** Server name: **DI Server** Hostname or IP address: **localhost** Port (empty is port 80): **9080** Web App Name (optional): **pentaho-di** Username: **admin** Password: **password** Is the master: **[unchecked]**
4	Click **[OK]**.
5	On the **View** tab, expand **Transformations** > **fact_sales_basic** > **Slave server**.
6	Right-click **DI Server** and choose **Monitor**.

Continued on next page

Guided Demo 11 – Scheduling & Monitoring, Continued

Monitoring a Slave Server, continued

Step	Action
7	When the transformation runs (according to the schedule), you should see detailed information on the **Slave server: DI Server** tab. To see the information, expand **Transformations** and click one of the **fact_sales_basic** entries in the view. 
8	Close the **Slave server: DI Server** tab.
9	Disconnect from the Repository.

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Guided Demo 12 – Detailed Logging throughout Execution

Introduction

In this guided demonstration you configure logging for job entries and steps.

Prerequisites

You must have Pentaho Data Integration (or Pentaho Business Analytics suite) installed and properly configured. You must also have access to course files required (if any).

Objectives

After completing this guided demonstration, you will be able to:

- Create database tables to hold logging information from jobs and transformations
- Configure logging for transformation steps and for job entries
- Examine logging information produced by jobs and transformations

Part I: Transformation logging

In part I of this guided demonstration, you configure logging for an existing transformation.

Transformation logging

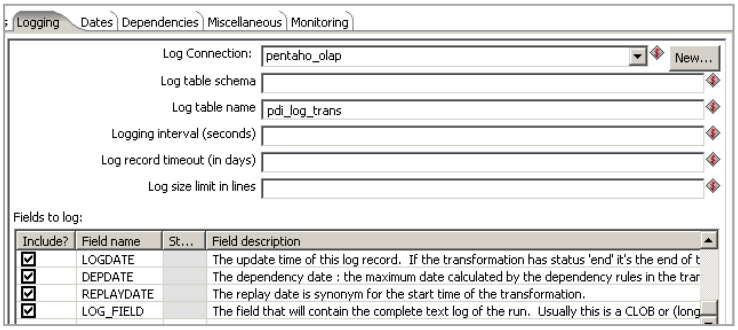
To configure logging for existing transformation - logging_trans1.ktr:

Step	Action
1	From the menu options, choose **File | Ope**n and navigate to: **`C:\pentahotraining\Support Files\GD12_Detailed Logging\`**
2	Open **`logging_trans1.ktr`**.
3	From the menu options, choose **Edit | Settings**.

Continued on next page

Guided Demo 12 – Detailed Logging throughout Execution, Continued

Transformation logging (continued)

Step	Action
4	Switch to the **Logging** tab and click **Transformation**.
5	In the **Transformation Properties** window, verify the following: **Field** / **Value** Log Connection: `pentaho_olap` Log table name: `pdi_log_trans` Also verify, in the **Fields to log** table, that the **Include** column is *checked* for **LOG_FIELD**.  **Warning**: Monitoring the LOG_FIELD field can negatively impact BI Server or DI Server performance. However, if you do not select all fields, including LOG_FIELD, when configuring transformation logging, you will not see information about this transformation in Enterprise Console.
6	Click **[SQL]** and then click **[Execute]**. This will create the **pdi_log_trans** table used in the exercise.
7	Close the results window and close the "Simple SQL editor."
8	Click **[OK]** to close the transformation properties.
9	Save your work and run the transformation.
10	Switch to the **View** tab.
11	Expand **Transformations > logging_trans1 > Database Connections**.
12	Right-click **pentaho_olap** and choose **Explore**.
13	In **Database Explorer**, expand **pentaho_olap > Tables**.
14	Right-click **pdi_log_trans** and choose **Preview first 100**.

Continued on next page

Guided Demo 12 – Detailed Logging throughout Execution, Continued

Transformation logging (continued)

Step	Action
15	You should see a single log entry for the transformation. 
16	Close the preview data window and close Database Explorer.
17	Switch to **`logging_trans1.ktr`** and run the transformation again.
18	Follow the previous steps to open the preview data window for the **`pdi_log_trans`** table. There should be a second row in the table with a new **`BATCH_ID`** representing the second execution of the transformation. 
19	Close the preview data window and close Database Explorer.
20	Leave **`logging_trans1.ktr`** open.

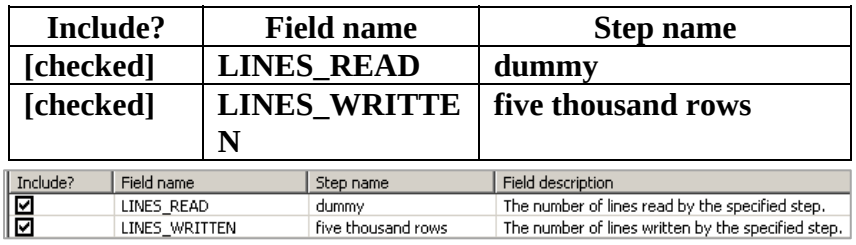
ID_BATCH	CHANNEL_ID	TRANSNAME	STATUS	LINES_READ	LINES_WRITTEN
0	12ea9eeb-c63e-44be-89ce-7c49cd6d867e	logging_trans1	end	5000	5000
1	82dfcee3-9ce1-4694-9d8b-7cd1101419e1	logging_trans1	end	5000	5000

Part II: Step logging

In part II of this guided demonstration, you configure logging for steps in the "logging_trans1.ktr" transformation.

Step logging

To configure step logging:

Step	Action
1	With **`logging_trans1.ktr`** open, choose **Edit | Settings** from the menu options.
2	Click the **Logging** tab.
3	Select **Transformation**.
4	In the **Fields to log** table, verify that the following fields are set: 

Include?	Field name	Step name
[checked]	**LINES_READ**	**dummy**
[checked]	**LINES_WRITTEN**	**five thousand rows**

Include?	Field name	Step name	Field description
☑	LINES_READ	dummy	The number of lines read by the specified step.
☑	LINES_WRITTEN	five thousand rows	The number of lines written by the specified step.

Continued on next page

Guided Demo 12 – Detailed Logging throughout Execution, Continued

Step logging (continued)

Step	Action
5	Click **[OK]** to close the "Transformation properties" dialog.
6	Save your work and run the transformation.
7	Switch to the **View** tab.
8	Expand **Transformations** > **logging_trans1** > **Database Connections**.
9	Right-click **pentaho_olap** and choose **Explore**.
10	In **Database Explorer**, expand **pentaho_olap** > **Tables**.
11	Right-click **pdi_log_trans** and choose **Preview first 100**.
12	In the preview data window, scroll to the **LOG_FIELD** column and view the data for the last line in the file (line 3). You should see log messages originating from the **dummy** and **five thousand rows** steps: **`2012/06/15 04:36:48 - five thousand rows.0 - Finished processing (I=0, O=0, R=0, W=5000, U=0, E=0)`** **`2012/06/15 04:36:48 - dummy.0 - Finished processing (I=0, O=0, R=5000, W=5000, U=0, E=0)`** 🕮 NOTE: You may copy the data in the LOG_FIELD column and paste it into a word processing document for readability.
13	Close the preview data window and close Database Explorer.
14	Close "logging_trans1.ktr."

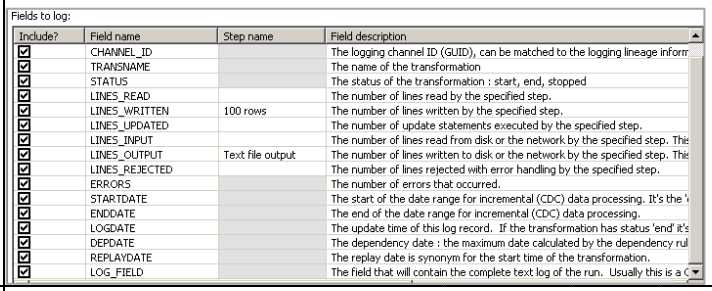
Part III: Transformation logging II

In part III of this guided demonstration, you configure logging for a second transformation using additional log settings.

Continued on next page

Guided Demo 12 – Detailed Logging throughout Execution, continued

Part III: Transformation logging II (continued)

Step	Action
1	From the menu options, choose **File | Ope**n and navigate to: **`C:\pentahotraining\Support Files\GD12_Detailed Logging\`**
	Open the **logging_trans2.ktr**
2	Switch to the **Logging** tab and click **Transformation**.
3	Verify the following settings and fields to log: ▪ **Log Connection**: **`Pentaho_olap`** ▪ **Log table name**: **`pdi_log_trans`** Include? / Field name / Step name [checked] LINES_WRITTEN 100 rows [checked] LINES_OUTPUT Text file output [checked] LOG_FIELD 
4	Click **[OK]** and run the transformation. If prompted, save your work.
5	Switch to the **View** tab.
6	Expand **Transformations > logging_trans2 > Database Connections**.
7	Right-click **pentaho_olap** and choose **Explore**.
8	In **Database Explorer**, expand **pentaho_olap > Tables**.
9	Right-click **pdi_log_trans** and choose **Preview first 100**.
10	In the preview data window, scroll to the **LOG_FIELD** column and view the data for the last line in the file (line 4 – TRANSNAME: logging_trans2). You should see log messages originating from the **Text file output** and **one hundred rows** steps: **`2012/08/02 14:49:45 - 100 rows.0 - Finished processing (I=0, O=0, R=0, W=100, U=0, E=0)`**

	2012/08/02 14:49:45 - Text file output.0 - Finished processing (I=0, O=101, R=100, W=100, U=0, E=0)

Continued on next page

Guided Demo 12 – Detailed Logging throughout Execution, Continued

Part III: Transformation logging II (continued)

Step	Action
13	Close the preview data window and close Database Explorer.
14	Close "logging_trans2.ktr."

Part IV: Job logging

In part IV of this guided demonstration, you configure logging for a PDI job.

Job logging

To configure job logging:

Step	Action
1	Open **`\pentahotraining\Support Files\GD12_Detailed Logging\logging_job.kjb`**.
2	From the menu options, choose **Edit** | **Settings**.
3	Click the **Log** tab.
4	Verify the following settings and fields to log: ▪ **Log Connection**: **`pentaho_olap`** ▪ **Log Table**: **`pdi_log_job`** **Enabled?** / **Field name** **`[checked]`** / **`LOG_FIELD`**
5	Click **SQL** and then click **Execute**.
6	Close the results dialog, close the "Simple SQL editor," and close the "Job properties" dialog.
7	Save your work and run the job.
8	Switch to the **View** tab.
9	Expand **Transformations** > **logging_job** > **Database Connections**.
10	Right-click **pentaho_olap** and choose **Explore**.
11	In **Database Explorer**, expand **pentaho_olap** > **Tables**.
12	Right-click **pdi_log_job** and choose **Preview first 100**.

Continued on next page

Guided Demo 12 – Detailed Logging throughout Execution, Continued

Job logging (continued)

Step	Action

13	In the preview data window, scroll to the **LOG_FIELD** column and view the data for the first line in the file. You should see log messages originating from the **dummy**, **five thousand rows**, **Text file output**, and **one hundred rows** steps:
	`2012/08/02 15:43:26 - five thousand rows.0 - Finished processing (I=0, O=0, R=0, W=5000, U=0, E=0)`
	`2012/08/02 15:43:26 - dummy.0 - Finished processing (I=0, O=0, R=5000, W=5000, U=0, E=0)`
	`2012/08/02 15:43:26 - 100 rows.0 - Finished processing (I=0, O=0, R=0, W=100, U=0, E=0)`
	`2012/08/02 15:43:26 - Text file output.0 - Finished processing (I=0, O=101, R=100, W=100, U=0, E=0)`

14	Close the preview data window and close Database Explorer.
15	Close "logging_job.kjb."
16	As time permits, test logging on your own by configuring logging for: ■ Steps ■ Performance ■ Logging channels

End of Guided Demonstration

Congratulations! You have completed this guided demonstration.

Course Completed!

Congratulations. You have completed the Pentaho Data Integration Fundamentals Course.

Appendix – Course Slides

pentaho

Pentaho Data Integration Fundamentals

Welcome!

PDI Version 5.2.1 Guide Version: 1.0

Module 1:

Introduction to Pentaho Data Integration

Lesson 1 – Objectives & Class Logistics

pentaho

Welcome Agenda

- Audience and prerequisites
- Learning objectives
- Class process
- Course outline

pentaho

Audience and Course Prerequisites

Intended audience :

- Technical users who integrate disparate data sources, build/maintain data models for analysis, and manage BI data/metadata, including:
  - Database Developers, Power Users, Technical Business Analysts, BI Solution Architects, Systems Integrators

Course prerequisites:

- None, but portions of the course assume knowledge of SQL and relational database concepts

pentaho

Learning Objectives

At the completion of this course, you should be able to:

- Describe the basic architecture of Pentaho Data Integration (PDI)
- Explain the role of PDI in the Pentaho Business Analytics Suite
- Load and write data to/from different data sources
- Join and merge data from different sources
- Use and Navigate the DI Repository
- Perform multiple data transformations
- Create calculations in a variety of methods
- Build portable and flexible jobs and transformations

pentaho

Introductions

Let us get acquainted with one another…

- Do you have any objectives for this course? If so, what are they?
- What are your expectations for this course?
- What is your current experience with PDI? (if any)
- Where are you located?

pentaho

Daily Schedule

Daily schedule:

- 9:00 am – 5:00 pm (10 Eastern for online courses)
- 1 hour lunch break (1 PM Eastern)
- Other breaks as needed

Times are approximate and the actual times for breaks and lunch is guided by the pace of the exercises and discussions.

Please ask questions to make this an interactive and fun learning experience!

pentaho

Course Structure

In general, most modules in this course follow this structure:

- Module
- Lesson
  - Demonstration(s)
  - Guided Demonstration(s)
  - Exercise(s)

Demonstrations are designed to have students watch the instructor introduce new topics.

Guided Demonstrations allow the student to have hands-on experience while still following along with the instructor.

Exercises give students the opportunity to work by themselves on a specific task.

pentaho

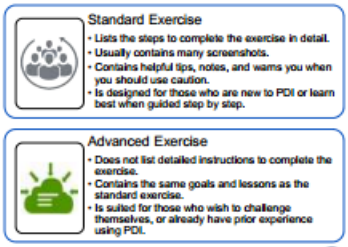
Course Modules

Module Name	Slide
Module 1: Introduction to Pentaho Data Integration	3
Module 2: Transformation Basics	27
Module 3: Reading & Writing Files	43
Module 4: Working with Databases	53
Module 5: Data Flow & Lookups	76
Module 6: Calculations	97
Module 7: Job Orchestration	105
Module 8: Exploring Data Integration Repositories	127
Module 9: Scheduling & Monitoring	135
Module 10: Detailed Logging	145

pentaho

Types of Exercises

In many of the modules in this course, students have a choice for the type of exercise they wish to complete. Here are the types of exercises found in this course:



pentaho

Verification of Student Environments

If you are using a Pentaho hosted environment…

Now we will connect to and examine the student environments…

pentaho

Module 1:

Introduction to Pentaho Data Integration

Lesson 2 – What is Pentaho Data Integration?

pentaho

Daily Schedule

Daily schedule:

- 9:00 am – 5:00 pm (10 Eastern for online courses)
- 1 hour lunch break (1 PM Eastern)
- Other breaks as needed

Times are approximate and the actual times for breaks and lunch is guided by the pace of the exercises and discussions.

Please ask questions to make this an interactive and fun learning experience!

Course Structure

In general, most modules in this course follow this structure:

- Module
- Lesson
  - Demonstration(s)
  - Guided Demonstration(s)
  - Exercise(s)

Demonstrations are designed to have students watch the instructor introduce new topics.

Guided Demonstrations allow the student to have hands-on experience while still following along with the instructor.

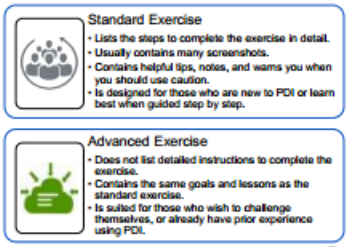
Exercises give students the opportunity to work by themselves on a specific task.

Course Modules

Module Name	Slide
Module 1: Introduction to Pentaho Data Integration	3
Module 2: Transformation Basics	27
Module 3: Reading & Writing Files	43
Module 4: Working with Databases	53
Module 5: Data Flow & Lookups	76
Module 6: Calculations	97
Module 7: Job Orchestration	105
Module 8: Exploring Data Integration Repositories	127
Module 9: Scheduling & Monitoring	135
Module 10: Detailed Logging	145

Types of Exercises

In many of the modules in this course, students have a choice for the type of exercise they wish to complete. Here are the types of exercises found in this course:



Verification of Student Environments

If you are using a Pentaho hosted environment…

Now we will connect to and examine the student environments…

Module 1:

Introduction to Pentaho Data Integration

Lesson 2 – What is Pentaho Data Integration?

Pentaho Mission

Enabling big data driven business

Modern cohesive business analytics and data integration platform

- Full spectrum of advanced analytics for all key roles
- Embeddable, cloud-ready analytics
- Big data blending for analytics in real-time environments
- Broadest and deepest big data integration

Innovation through open source

- Open, pluggable, purpose built for the future
- Early sustained leadership in big data ecosystem with technology innovation

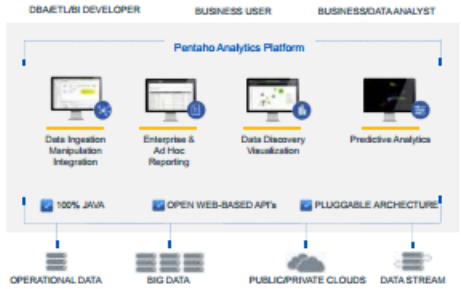
Critical mass achieved

- Over 1,200 commercial customers
- Over 10,000 production deployments

pentaho

Pentaho: Platform Overview

No Boundaries – Open, Extensible, Simplified Platform



pentaho

Data Integration

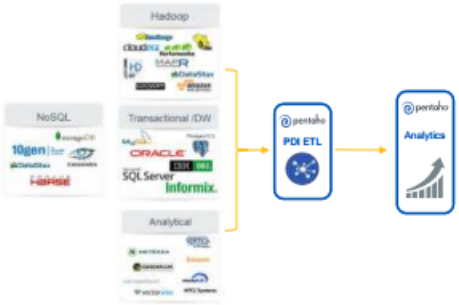
ETL, Scheduling, Events, Orchestration

- 100% Java engine
- Meta-data driven architecture – graphical ETL Designer
- Scale-out architecture, deployable to
  - Desktop
  - PDI clusters
  - Hadoop clusters
- Plugin architecture for extensibility
- Batch, low-latency and real time processing
- Rapid onboarding of Analytics
- Embeddable

pentaho

Broad Connectivity

Broad connectivity combined with powerful data integration



pentaho

PDI Development Concepts

pentaho

Common Uses

Data warehouse population:

- Built-in support for slowly changing dimensions, junk dimensions and other data warehouse concepts

Export of database(s) to text-file(s) or other databases

Import of data into databases, ranging from text-files to Excel® spreadsheets

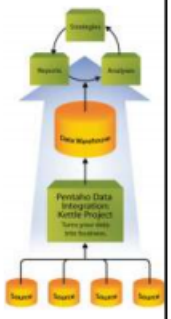
Data migration between database applications

Exploration of data in existing database (tables, views, synonyms)

Information enrichment by looking up data in various information stores (databases, text files, spreadsheets)

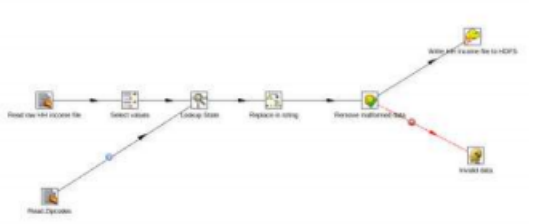
Data cleansing by applying complex conditions in data transformations

Application integration



pentaho

Concept – Data Transformations



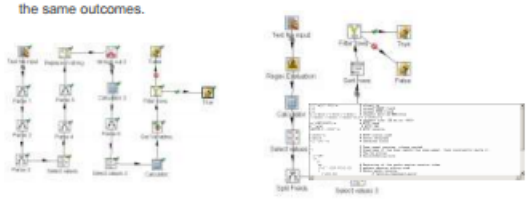
INPUT(S) – PROCESS(ES) – OUTPUT(S)

Concept – Jobs (orchestrate)

START* – CHECK – WATCH – EXECUTE – NOTIFY - *FINISH

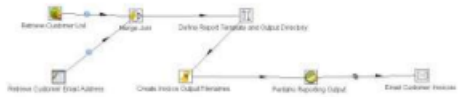
Concept - Visual Development

- Ability to build a solution through a GUI interface requiring a no coding approach
- Flexibility to use a scripting or coding approach to leverage existing code or by preference.
- Two examples below show parsing a web log through a no code approach and through using a regular expression scripting approach. Both methods produce the same outcomes.

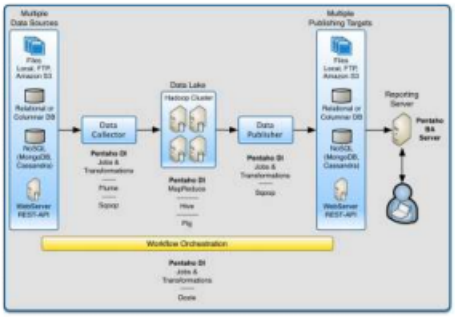


Concept – Report Bursting

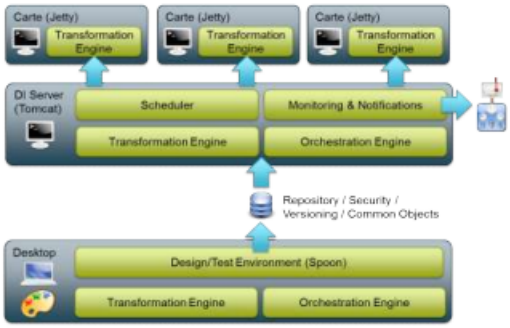
- Retrieve data from any source
- Merge data with any other source
- Inject data into an Operational Report
  - Slice data into aggregates (eliminating the need for multiple source passes)
- Archive each Report
- Distribute the Report to each intended Recipient, via Email, FTP, Etc.



Pentaho Architecture



Architecture



Pentaho Data Integration Components

Spoon

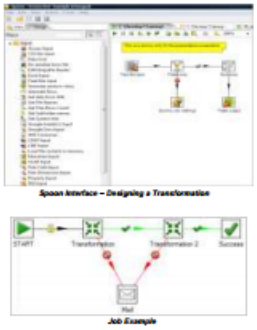
- Graphical environment for modeling
- Transformations are metadata models describing the flow of data
- Jobs are workflow-like models for coordinating resources, execution, and dependencies of ETL activities

Pan

- Command line tool for executing transformations modeled in Spoon

Kitchen

- Command line tool for executing jobs modeled in Spoon

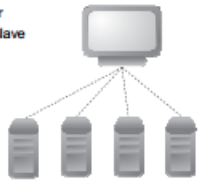


pentaho

Pentaho Data Integration Components

Carte

- Lightweight web\HTTP server for remotely executing jobs and transformations
- Carte accepts XML containing the transformation to execute and the configuration
- Enables remote monitoring
- Used for executing transformations in a cluster
- Remote servers running Carte referred to as slave servers



pentaho

Module 2:

Transformation Basics

Lesson 1 – Learning the PDI User Interface

Demo 1 – Explore PDI's Interface

pentaho

Starting the user interface

Launch `spoon.bat` (Windows) or `spoon.sh` (Linux, MacOS) in `design-tools/data-integration`.

The command `launch-designer.bat /.sh` is used in the archive-based installation.

In this portion of the course, you do not use the repository.

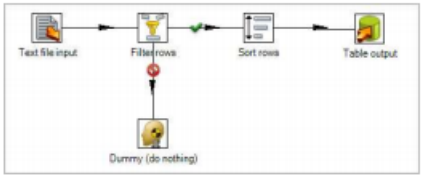
- All training data will be stored in the file system as `.KTR` (transformations) or `.KJB` files (jobs) in XML format.

pentaho

Transformations

Transformations are a network of logical tasks (steps):

- Read a flat file
- Filter it based on a condition
- Sort it
- Load it into MySQL



pentaho

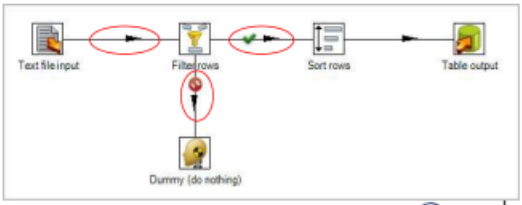
Hops - Within Transformations

Are data pathways that connect steps together

Allow schema metadata to be passed from step to step

Determine the flow of data through the steps

Example: The pathway for all data and the true and false paths from a Filter rows step



pentaho

Values, Metadata and Data

Values are like columns in data rows.

- They are composed of the metadata and the data.
- Metadata is only transported with the first data row.
- All subsequent data rows reference the metadata.

PDI maps database (JDBC) data types to PDI data.

- Implementation can (and often is) different from database to database.

pentaho

Values, Metadata and Data

Metadata is used for formatting when:

- Data is presented (in a preview)
- Data is written to the outside world (to a text or XML file)

Metadata is NOT used for formatting when:

- Data is just loaded from one table and written to another table

Metadata is used to create SQL-statements with field types, length, and so on.

Metadata is used to check the data for the correct data type.

Note - A change in metadata does not change the data:

- Modification of length does not change/truncate data
- New formatting does not change data

pentaho

Data Types

- Number (double, floating point)
- Integer
- BigNumber
- String
- Date – Includes date and time. yyyy/MM/dd HH:mm:ss.SSS
- Boolean – True/False
- Binary – Mainly used for Blob data.
- Serializaible – An object to transfer from/to specific steps
- Plug-in
- Note: Formatting Number Data Types is done by default with the pattern #.#;-#.# (only one digit precision)

pentaho

Samples

PDI comes with a large variety of samples to help you learn and understand the various ways of using its features.

Here are some examples of the types of samples included:

- Transformations
- Jobs
- Mappings
- Various types of files to be used as input (e.g. Txt, CSV, JSON)

pentaho

Guided Demo 1

Launch & Customize PDI

Main tree

- Lists all open transformations and jobs and their contents

Core objects and favorite steps/job entries

- Core is toolbox with all the available steps/job entries (plug-ins are in bold)
- Favorites are static 'most' used steps

Options and settings

- Options are valid for the entire PDI environement
- Settings are valid for a particular transformation or job

When completing any guided demonstration or exercise, the specified folder locations and filenames for objects that you create are suggestions. You may use any location or filename that you like, noting the change for your future reference.

pentaho

Module 2:

Transformation Basics

Lesson 2 – Creating Transformations

pentaho

Guided Demo 2
Creating a 'Hello World' Transformation

- Generate Rows
- Outputs certain number of rows
- Default is empty but can contain a set of static fields
- Output options:
  - **Limit**: The number of rows to output
  - **Fields**: Static fields user includes in output row

pentaho

Exercise 1
Generate Rows, Sequence, Select Values

Time: 25 Minutes

Overview: In this exercise, you will create a transformation and separately add and configure each step and connecting hops. You will preview each step to ensure what is previewed is what is expected.

Objectives: After completing this exercise, you will be able to:

- Learn to create a new transformation.
- Add steps and hops.
- Configure and preview steps
- Execute the transformation.

pentaho

Module 2:
Transformation Basics

Lesson 3 – An Introduction to Error Handling & Logging

TIP: If you have trouble launching Spoon, try launching it via SpoonDebug.bat. It is a troubleshooting utility that logs Spoon's console messages and other statistics to <pentaho_install_dir>\design-tools\data-integration\SpoonDebug.txt

pentaho

Guided Demo 3
Error Handling & Basic Logging

Time: 20 Minutes

Overview: This guided demonstration provides an introduction to finding and handling errors, as well as basic logging.

Objectives: After completing this exercise, you will be able to:

- Know when you have an error.
- Find the log file and specify to view only the errors.
- Understand how to use the Execution Pane's Step Metrics and Logging tabs.

pentaho

Module 2:
Transformation Basics

Lesson 4 – Introduction to Repositories

pentaho

Guided Demo 4
Saving a Transformation in the Repository

Time: 20 minutes

Objective: Learn how to save your transformations in the repository using an existing repository connection.

Objectives:

- Log into and examine the repository.
- Save an object in the repository.
- Export and Save to the file system when currently connected to the repository.

pentaho

Guided Demo 2
Creating a 'Hello World' Transformation

- Generate Rows
- Outputs certain number of rows
- Default is empty but can contain a set of static fields
- Output options:
  - **Limit**: The number of rows to output
  - **Fields**: Static fields user includes in output row

pentaho

Exercise 1
Generate Rows, Sequence, Select Values

Time: 25 Minutes

Overview: In this exercise, you will create a transformation and separately add and configure each step and connecting hops. You will preview each step to ensure what is previewed is what is expected.

Objectives: After completing this exercise, you will be able to:

- Learn to create a new transformation.
- Add steps and hops.
- Configure and preview steps
- Execute the transformation.

pentaho

Module 2:
Transformation Basics

Lesson 3 – An Introduction to Error Handling & Logging

TIP: If you have trouble launching Spoon, try launching it via SpoonDebug.bat. It is a troubleshooting utility that logs Spoon's console messages and other statistics to <pentaho_install_dir>\design-tools\data-integration\SpoonDebug.txt

pentaho

Guided Demo 3
Error Handling & Basic Logging

Time: 20 Minutes

Overview: This guided demonstration provides an introduction to finding and handling errors, as well as basic logging.

Objectives: After completing this exercise, you will be able to:

- Know when you have an error.
- Find the log file and specify to view only the errors.
- Understand how to use the Execution Pane's Step Metrics and Logging tabs.

pentaho

Module 2:
Transformation Basics

Lesson 4 – Introduction to Repositories

pentaho

Guided Demo 4
Saving a Transformation in the Repository

Time: 20 minutes

Objective: Learn how to save your transformations in the repository using an existing repository connection.

Objectives:

- Log into and examine the repository.
- Save an object in the repository.
- Export and Save to the file system when currently connected to the repository.

pentaho

Module 3:
Reading & Writing Files

Lesson 1 – Input & Output Steps

pentaho

A Selection of Input and Output Steps

pentaho

Demonstrations 2-5

Demonstrations of the most often used Input & Output steps.

Discussion of the options.

- Browse
- Get Fields
- Content information

KTR Slides

pentaho

Guided Demo 5

Combining Several Inputs into One Output

Time: 20 Minutes

Objective: Read multiple data files, add a system date and output to one file.

pentaho

Module 3:
Reading & Writing Files

Lesson 2 – Parameters & kettle.properties

pentaho

Setting Environment Variables: Properties File

You can set environment variables in the `kettle.properties` properties file.

- By default, **`kettle.properties`** is located in your **`$HOME/.kettle`** directory (for example, **`\Users\jb\.kettle`**):

```
# This file was generated by Pentaho Data Integration version 3.0.0-RC2.
#
# Here are a few examples of variables to set:
#
# PRODUCTION_SERVER = hercules
# TEST_SERVER = zeus
# DEVELOPMENT_SERVER = thor
#
# Note: lines like these with a # in front of it are comments
#
```

You can also point the `KETTLE_HOME` environment variable to the directory that contains the `.kettle` directory.

- See the PDI Overview module for additional details.

pentaho

Guided Demo 6

Creating kettle.properties Parameters

Time: 15 Minutes

Objective: Create two kettle.properties parameters that are to be used for referencing the file path used for your input and output files.

- `DIR_INPUT = C:\pentahotraining\DataFiles\Input`
- `DIR_OUTPUT = C:\pentahotraining\DataFiles\Output`

These parameters will be used throughout the remaining modules of this course.

pentaho

Exercise 2

CSV Input to Multiple Text Output Using Switch/Case

Time: 30 Minutes

Overview: In this exercise, you will create a transformation that reads a CSV file containing order and country of origin data. Then, it will send incoming data for specific country's orders.

Objectives:
After completing this exercise, you will be able to:

- Create and use transformation parameters to define locations for input and output folder locations.
- Create and configure a CSV file input step
- Create and configure a Switch / Case step that will send data to specific steps depending on the data contained in the incoming data stream.
- Create and configure a Text file output step that will create a new text file
- Create hops connected from a Switch / Case step that are configured to specific case values.

pentaho

Exercise 3

Serializing Multiple Text Files

Time: 10 Minutes

Overview: In this exercise, you will create a transformation that reads the multiple text files you created in the previous exercise. Then, the data is sent to a serialize step that creates a binary (serialized) file. A regular expression is used, along with a parameter, to determine the correct files to read.

Objectives: After completing this exercise, you will be able to:

- Read multiple text files from a parameterized directory using the Text file input step.
- Use a regular expression to specify a specific pattern of filename.
- Create and configure a Serialize step to create binary file from an input stream.

pentaho

Exercise 4

De-serializing a File

Time: 10 Minutes

Overview: In this exercise, you will create a transformation that reads serialized file you created in the previous exercise and de-serializes. Then, the data is sent to a **Text file output step.**

Objectives: After completing this exercise, you will be able to:

- De-serialize a file.
- Create a text file output with minimal width delimited data and customized fields.

pentaho

Module 4:

Working with Databases

With an introduction to the training data.

Lesson 1 – Connecting to & Exploring a Database

pentaho

Introduction to the Training Data

Represents fictitious company: Steel Wheels

- Buys collectable model cars, trains, trucks, and so on from manufacturers
- Sells to distributors across the globe

Data adapted from sample data provided by Eclipse BIRT project

pentaho_oltp database has many tables:

- Offices, Employees, Customers, Products, Orders, Orderdetails, Payments, and so on

pentaho_oltp

pentaho

Pentaho Training Data: Tables (1 of 3)

Offices

- 7 offices worldwide (San Francisco, Boston, NYC, Paris, Tokyo, Sydney, London)
- Headquartered in San Francisco, CA
- Each office assigned to a sales territory (APAC, NA, EMEA or JAPAN)

Employees

- 23 employees: 6 executives and 17 sales representatives
- Each assigned to one of seven offices
- Sales representatives also assigned to particular number of customers (distributors)
- New sales representatives (still in training) do not have assigned customers

pentaho

Pentaho Training Data: Tables (2 of 3)

Customers

- Steel Wheels has 122 customers worldwide
- Approximately 20 are new customers without a sales representative
- Each has a credit limit which determines maximum outstanding balance

Products

- 110 unique models purchased from 13 vendors
- Classified as 7 distinct product lines: Classic cars, vintage cars, motorcycles, trucks and buses, planes, ships, trains
- Models classified based on scale (1:18, 1:72, and so on)
- Cost paid and MSRP (manufacturer's suggested retail price)

pentaho

Pentaho Training Data: Tables (3 of 3)

Orders

- 2,560 orders spanning period from 1/1/2000 to 12/31/2007
- Each in a given state: In process, shipped, cancelled, disputed, resolved, on hold

Orderdetails

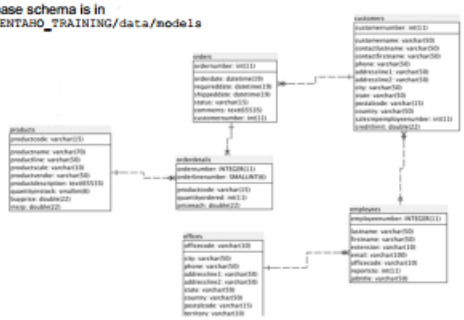
- Order line items reflect negotiated price and quantity per product
- Training database has 23,640 records in Orderdetails

pentaho_oltp

pentaho

Database Schema OLTP (Source Database)

Database schema is in
$PENTAHO_TRAINING/data/models



pentaho

Database Connections

Multiple, shared connections to different databases can be created.

When using a DI Enterprise repository:

- Defined connections are readily available to transformations and jobs.

When not using a DI Enterprise repository:

- Connection definition is contained in a single transformation or job.
- Connection definitions can be shared in subsequent transformations and jobs.

The Connection information for the repository is stored in repositories.xml. It is located in the user's home folder at, ~/.kettle/repositories.xml.

pentaho

Access Via JDBC

Database drivers can be added to the /data-integration/lib.

- Use Generic tab of connection dialog to use unlisted drivers
- Permits connections to non-listed databases

Existing drivers can be replaced in the /data-integration/lib directory.

pentaho

Other Access Methods

ODBC connections are possible:

- ODBC connections must be defined in Windows.
- ODBC connections made via ODBC-JDBC-Bridge.
  - Some limitations on SQL syntax
  - Generally slower than JDBC due to additional layer

Use a JNDI connection to connect to a data source defined in an application server like JBoss or WebSphere.

Plugin specific access methods are supplied by a specific database driver (like SAP R/3 or PALO connections).

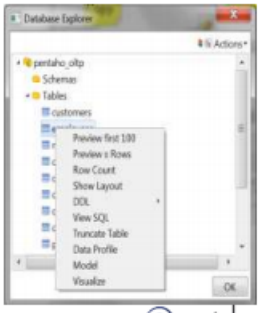
pentaho

Database Explorer

Allows users to explore database tables.

Found on the toolbar:

And in the connection context menu:



pentaho

Guided Demo 7

Connections & the Database Explorer

Time: 35 minutes

Objective: Learn how to use the various database-related features of Spoon. Including those shown below.

Create a Database Connection to sampledata

Use Database Explorer to explore the connection

Use the Simple SQL Editor

pentaho

Module 4:

Working with Databases

Lesson 2 – Table Input & Output Steps

pentaho

Demo 6 – Reading a Database Table
Demo 7 – Loading a Database Table

pentaho

Exercise 5

Reading & Writing to Database Tables

Time: 35 Minutes

Overview:

- This exercise is designed to introduce you to various methods of interacting with data using Pentaho Data Integration.
- In this exercise, you create a database connection, explore a data source, and create transformations that use various data input and output steps.

Objectives:

- Create a database connection
- Use Database Explorer to interact with a data source
- Create a transformation that uses the Table input and Table output steps
- Create a transformation that uses the Text file output step

pentaho

Module 4:
Working with Databases

Lesson 3 – Insert / Update, & Delete Steps

pentaho

Demo 8 - Examine Insert, Update, & Delete Steps

KTR Slide

pentaho

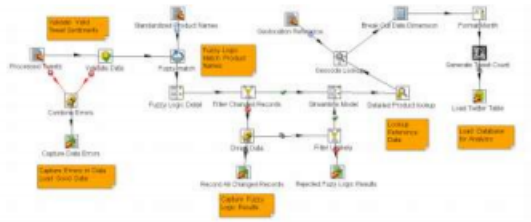
Module 4:
Working with Databases

Lesson 4 – Data Cleansing

pentaho

Concept - Data Cleansing

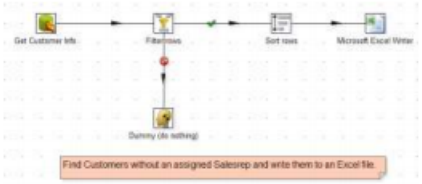
- Validate data by data type, expected values, ranges, etc.
- Normalize data based on defined data definitions using Fuzzy Logic
- Capture all data abnormalities for further processing
- Enrich validated data



pentaho

Guided Demo 8
Data Cleansing

Create a transformation to correct issue with customer data where they have no assigned sales representative. The records will be output to a newly created Excel file. The Filter rows and Sort rows steps are introduced.



pentaho

Module 4:
Working with Databases

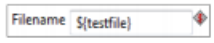
Lesson 5 - Using Parameters & Arguments in SQL

pentaho

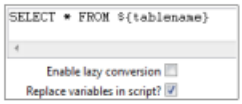
Referencing Variables

Variables are very useful in:

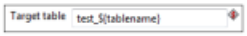
- Flexible file processing



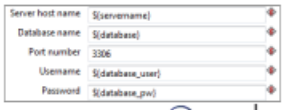
- Flexible table processing
  - Table input:



  - Table output:



- Flexible database connections



pentaho

Demo 9 – Arguments in SQL
Demo 10 – Parameters in SQL

The instructor will explain how to use arguments and parameters almost anywhere in Spoon where SQL is used.

pentaho

Exercise 6

Input with Parameters / Table Wizard

Time: 35 Minutes

Overview:

- This exercise is designed to introduce you to various methods of updating and deleting data using Pentaho Data Integration.

Objectives:

- Create a transformation that uses the CSV file input and Insert/Update steps
- Create a transformation that uses the Table input step that loads data based on a parameter value
- Use the Copy Table wizard

pentaho

Module 5:

Data Flow & Lookups

Lesson 1 – Copying & Distributing Data

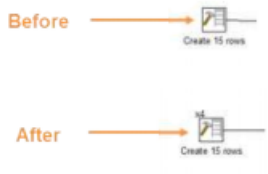
TIP: To optimize performance of a transformation, start with the slowest running step, it determines the speed of a transformation.

pentaho

Demo 11

Starting Multiple Step Instances

The instructor will explain how to and the best time for starting multiple instances of a step.



pentaho

Data Flow, Threading Mechanism

All steps are started and run in parallel.

- Initialization sequence is not predictable

PDI manages the correct data flow.

- 'Pulling and pushing' data from step to step

PDI can process an unlimited number of rows:

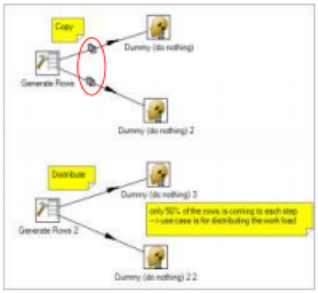
- Steps vary on execution speed and memory consumption
- Set a threshold on number of rows and next step will wait
  - If number of rows is reached, source step waits for room
  - When there is room, more rows are put into data stream
  - Transformation Properties / Miscellaneous / Number of rows in rowset

pentaho

Copying or Distributing Data

Specify if data can be copied or distributed to multiple target hops (right-click a step and select 'Data Movement').

Note the icons for copy.



pentaho

Interpreting Runtime Data

pentaho

Runtime Data

Tabular data that helps a developer understand information about the transformation as it runs.

- Information about "Number of Records" streaming
- Time
- Records/second
- Status of steps
- Input/output records on hops

pentaho

Basics

"Log" view when running transformations

Columns:

- **Stepname**: Name of the step (lookup_region, read_orders)
- **Copynr**: Copy numner if multiple copies are started (0,1,2,3,4, ...)
- **Read**: Number of records received from PREVIOUS step
- **Written**: Number of records passed to NEXT step
- **Input**: Number of records read from a file, database, and so on
- **Output**: Number of records output to a file, database, and so on
- **Rejected**: Number of records rejected
- **Errors**: Number of errors
- **Active**: Status of step (initializing, running, finished, and so on)

Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active
Step1	0	0	11466	11466	0	0	0	0	Running
Step2	0	1466	1465	0	0	0	0	0	Running
Step3	0	1465	1465	0	0	0	0	0	Running

pentaho

Example

11468 read from database

Read 1466 records

Read 1465 records

Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active
Step1	0	0	11466	11466	0	0	0	0	Running
Step2	0	1466	1465	0	0	0	0	0	Running
Step3	0	1465	1465	0	0	0	0	0	Running

pentaho

Time and Records

Time

- Number of seconds from start of step to either:
  - Now() if still running
  - Time of last recorded if finished

Time	Speed (r/s)
327.0	72.2
575.1	41.1
575.1	41.1

Speed

- Number of records/time
- Time follows above formula to show:
  - Live throughput while running
  - Aggregate throughput for entire run if finished
- **Note**: The "speed" of a step is not solely dependent on the step itself. This is clarified on the following pages.

pentaho

Input / Output Interpretation

Where "rows" are sitting on hops gives you information for improving performance.

Formal:

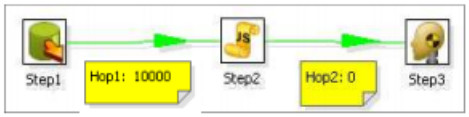
- Look for the latest downstream step with few records on OUTPUT and many records on INPUT

Informal:

- Look for the first "0" on OUTPUT and "10000" on INPUT

Step3 is processing rows as fast as Step2 produces

Step1 is producing rows faster than Step2 consumes



pentaho

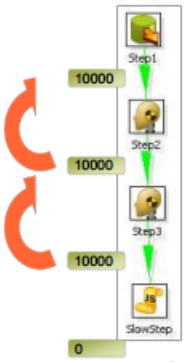
Backlog

Slow steps cause a backlog.

Look at input/output for all the steps.

See what is causing the backlog (downstream with 0 on the output)

NOT just what is backed up.



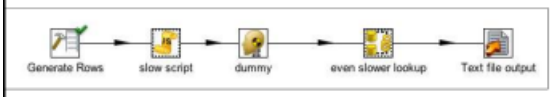
pentaho

Viewing Bottlenecks

- Check the rows/second metric of each highlighted step to see if processing speed is reasonable.
- Sometimes steps have a slightly tilted input/output ratio, but perform blazingly fast in absolute terms!

A high input/output buffer ratio for a step is represented by a dotted line.

- These steps receive rows faster than they can process them (bottleneck steps).
- These are often relatively slow scripting steps or I/O bound steps.



pentaho

Exercise 7

Parallel Processing

Time: 30 Minutes

Overview: In this exercise, you will learn how to create and then execute more than one instance of a step at the same time.

Objectives:

- Copy and Distribute data
- Start multiple copies of a step
- Send data to multiple outputs

pentaho

Module 5:

Data Flow & Lookups

Lesson 2 – Lookups

pentaho

Lookups

The Lookup feature in PDI accesses a data source to find values using a defined matching criteria (a key value).

The following commonly used steps have lookup functionality:

- Database lookup
- Stream lookup
- Merge join

Other steps have lookup functionality as well:

- Database join
- Dimension lookup / update
- Combination lookup / update
- HTTP lookup

pentaho

Merge Join

Merge Join

Merge join step performs a merge join between data sets using data from two different input steps.

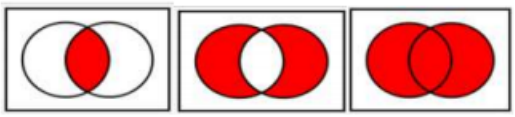
- Join options: INNER, LEFT OUTER, RIGHT OUTER, and FULL OUTER

- **Options include:**
  - **Step name:** Unique name of step
  - **First Step:** First input step to the merge join
  - **Second Step:** Second input step to the merge join
  - **Join Type:** INNER, LEFT OUTER, RIGHT OUTER, or FULL OUTER
  - **Keys for 1st step:** Key fields on which incoming data is sorted
  - **Keys for 2nd step:** Key fields on which incoming data is sorted

pentaho

Merge Join

- INNER = Only output a row when the key is in both streams
- LEFT OUTER = Output a row even if there is no matching key in 2nd step
- RIGHT OUTER = Output a row even if there is no matching key in 1st step
- FULL OUTER = Output a row regardless of matching



Inner **Right & Left Outer** **Full Outer**

(Both Shown Together Above)

pentaho

Merge Rows Diff

Merge Rows

Compares and merges two streams of data: Reference stream and compare stream

Mostly used to identify deltas in source data when no timestamp is available

- Reference stream = Previously loaded data
- Compare stream = Newly extracted source data

Usage note: Ensure streams are sorted by comparison key fields

The output row is marked as follows:

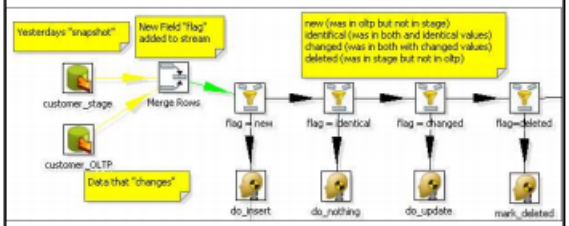
- **Identical:** Key found in both streams and compared values were identical
- **Changed:** Key found in both streams but one or more values is different
- **New:** Key not found in the reference stream
- **Deleted:** Key not found in the compare stream

pentaho

Pattern: Change Data Capture

Compare previous data to current data

Route to appropriate processing



pentaho

Guided Demo 9

Choosing Adequate Sample Size for 'Get Fields'

Time: 20 minutes

Description:

In this guided demo, you will witness the importance of choosing an adequate number of preview rows for input steps.

Objectives:

- Properly choose a sample size for a CSV file input step. The same technique is used for a Text file input step as well.
- Identify transformation errors whose cause may be an improper sample size chosen in a previous step.

pentaho

Exercise 8

Lookups & Data Formatting

Time: 30 Minutes

Objectives: This exercise introduces the concept of looking up data. The exercise scenario includes a flat file (.csv) of sales data that you will load into a database so that mailing lists can be generated. Several of the customer records are missing postal codes (zip codes) that must be resolved before loading into the database.

Tasks:

- After completing this exercise, you will be able to:
  - Merge Data from different streams.
  - Use the Select Values step to change the name and format of a field.

pentaho

Module 6:
Calculations

Lesson 1 – Using the Group By Step

pentaho

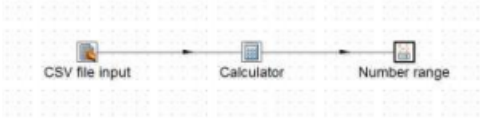
Calculations and Scripting

pentaho

Guided Demo 10

Creating Summary Fields Using Group By

Calculate the Elapse hours between Shipped date and Order Date.



pentaho

Formula Step

Formula

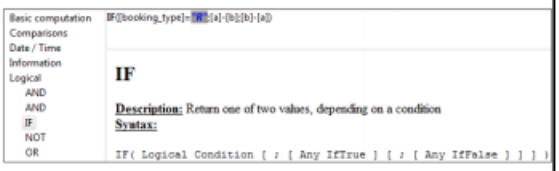
The Formula step is based on the Oasis OpenFormula syntax.

- See: https://www.oasis-open.org/

You can reference values using square brackets: `[value]`

Help is available for every function.

Applying business rules (if / then / else) with more complex logic is possible in a Formula step.



pentaho

Regex Evaluation

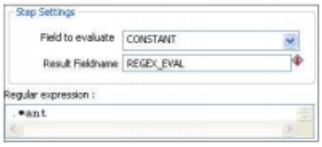
Regex Evaluation

Evaluates a Regular Expression

Options include:

- **Field to Evaluate:** Name of EXISTING field containing string you want to evaluate
- **Result Fieldname:** Name of the NEW field to hold result (Y/N value)
- **Regular Expression:** Regular expression to evaluate

- Other Options: Case Sensitivity, Encodings, Whitespace, Capturing Groups, and so on



pentaho

User Defined Java Expressions

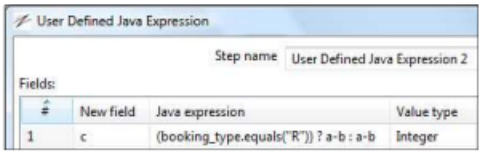
User Defined Java Expression

The User Defined Java Expression step compiles into pure Java.

- You may use standard Java object methods for values.

Applying business rules (if / then / else) with more complex logic is possible using conditional expression operators:

- (a > b) ? a : b
- The condition *(a > b)* is tested. If it is true, the first value *a* is returned. If it is false, the second value *b* is returned.
- Another example that tests a string value:



pentaho

User Defined Java Class

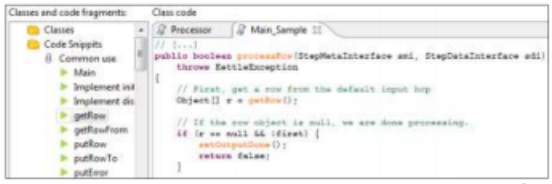
User Defined Java Class

The User Defined Java Class step allows access all internal step logic (similar to a custom plug-in).

- The benefit of User Defined Java Class is to simplify the deployment process.

Further information can be found on the Wiki:
http://wiki.pentaho.com/display/EAI/Writing+your+own+Pentaho+Data+Integration+Plug-In

Code snippets provide samples:



pentaho

Exercise 9

Calculating & Aggregating Order Qty

Time : 20 Minutes

Objectives:

- In this exercise, you will create a transformation that reads the CSV file containing order data. Then, the data is sorted, by country and grouped by country on the total quantity ordered. Finally, the percentage of the quantity ordered is calculated..

Tasks:

- After completing this exercise, you will be able to:
  - Sort rows of data on the stream.
  - Group rows
  - Use the Calculator step to calculate values based on values of data in the stream.

pentaho

Module 7:

Jobs Orchestration

Lesson 1 – Introduction to Jobs

pentaho

What is Job Orchestration?

In an IT environment, orchestration is the ongoing task of ensuring that technical processes run reliably and according to schedule while producing the correct outputs using acceptable amounts of resources

Orchestration includes:

- Ensuring the availability of resources
- Executing a series of tasks in the right order
- Detecting and recovering from errors
- Logging the results
- Notifying people and other applications

pentaho

Common Causes for Transformation Execution Failure

- An external system is unavailable
- A source data file gets corrupted
- A target database becomes full
- A shared resource gets busy
- Someone changes a password
- Another developer changes their code
- The power goes out
- A transformation fails
- A Hadoop job fails
- And so on

pentaho

Defense is the Best Practice

Always check the status of external systems

- Host systems
- Databases / tables
- File systems / directories
- FTP sites
- Web services

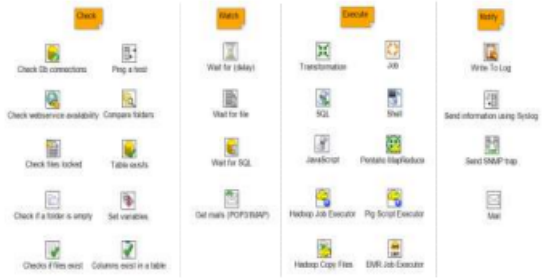
Execute tasks with retry logic

Notify others when a problem has occurred

Store variables in a central location and use them consistently

pentaho

PDI Job Entry Toolkit



Simple Example

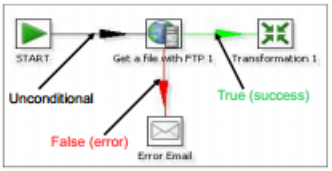
Job Hops

A job hop is a graphical representation of one or more data streams between two steps.

A hop always links two job entries and can be set (depending on the type of source job entry) to execute the next job entry:

- Unconditionally
- After successful execution
- Or after failed execution

The execution order is indicated by a coloured arrow in the graphical view:

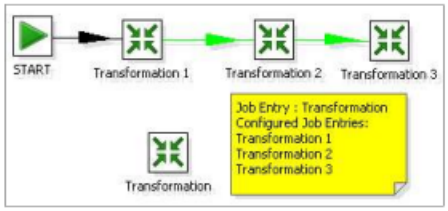


Job Entry

A job entry is the primary building block of a job.

- Execute transformations, retrieve files, generate email, and so on

A single job entry can be placed on the canvas multiple times.



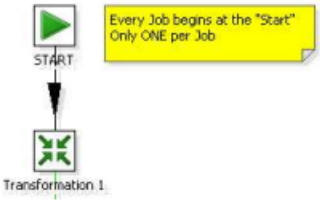
General Job Entries

Start

Start defines the starting point for job execution.

Only unconditional job hops are available from a start job entry.

The Data Integration Server provides a more robust option for scheduling execution of jobs and transformations and is the preferred alternative to scheduling using the Start step. If you want the job to run like a daemon process, however, enable Repeat in the job settings dialog box.



Transformation

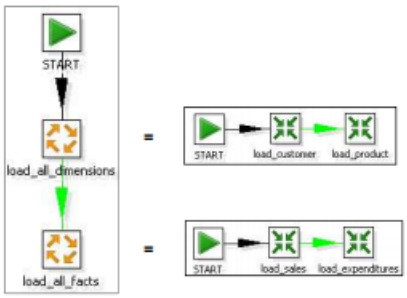
The Transformation step executes a transformation.

Options include:

- **Name of the job entry**: Unique job name
- **Name of transformation**: Name of the transformation to load
- **Repository directory**: Repository directory where transformation is located
- **Filename**: XML file name of the transformation to load
- **Specify log file**: Specifies a separate logging file for the execution of this transformation
- **Name of log file**: Directory and name of the log file (for example: `C:\logs`)
- **Extension of the log file**: File name extension (for example: log or txt)

pentaho

Job (Sub-Job) Example

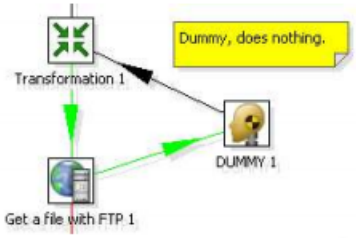


pentaho

Dummy

Dummy

Use the dummy job entry to do nothing in a job.
This can be useful to make job drawings clearer or for looping.
Dummy performs no evaluation.



pentaho

Scripting Job Entries

Executes a script on the host where the job is running.

pentaho

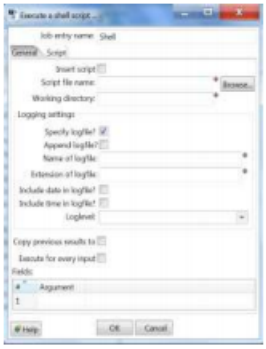
Shell

Shell

Executes a shell script on the host where the job is running.

Copy and paste the script into the Script tab or insert the script by clicking the Insert script box

Specify logging settings and pass arguments.



pentaho

SQL

Execute an SQL script.

You can execute more than one SQL statement, provided that they are separated by semi-colons.

The options for this job entry are:

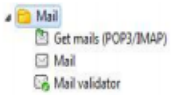
- **Connection**: The database connection to use.
- **SQL script**: The SQL script to execute.

Example

pentaho

Mail Categories

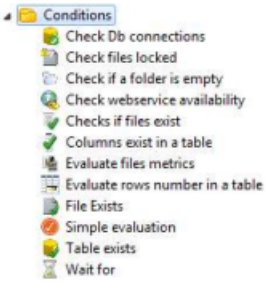
Use the **Mail** job entry to send a text or HTML email with optional file attachments, to validate a mail address or to pull mail from a POP server.



pentaho

Conditions Category

Pulls information from folders, files, columns and tables.



pentaho

Table Exists

Use the **Table exists** job entry to verify that a specified table exists on a database.

Pentaho Data Integration returns a True or False value depending on whether or not the table exists.

The options provided for this job entry are:

- **Database connection**: The database connection to use.
- **Table name**: The name of the database table to check. .

Note: This job entry performs one check and then moves on. If you want to poll until the tables appear, use the Wait for File or Wait for SQL job entries which have a polling interval parameter.

pentaho

File Exists

Use the **File exists** job entry to verify that a specified file exists on the server on which Pentaho Data Integration is running.

Pentaho Data Integration returns a True or False value depending on whether or not the file exists.

The options provided for this job entry are:

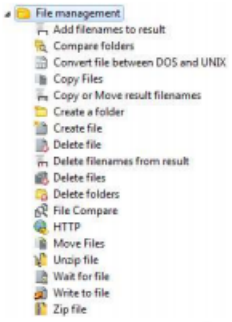
- **Filename**: The name and path of the file to verify.
- **Variable**: Select the variable to use as filename.
- **Browse**: Look for the file on the file system.

Note: This job entry performs one check and then moves on. If you want to poll until the files appear, use the **Wait for File** or **Wait for SQL** job entries which have a polling interval parameter.

pentaho

File Management Category

Housekeeping job entries



pentaho

Exercise 10

Loading JVM Data into a Table

Time: 45 Minutes

Objectives: Monitoring memory is an important part of making sure applications written in Java, like PDI, execute efficiently. In this exercise, you create a job that runs an existing transformation that reads the Java Virtual Memory (JVM) heap data and loads it into a database table. You define the database table's name as a PDI environmental variable

Tasks:

- Open and understand one of the sample transformations that come with PDI.
- Set PDI environmental variables.
- Create a new job and configure it to execute a transformation.
- Execute a job.

pentaho

Module 8:
Exploring Data Integration Repositories

Lesson 1 – The Pentaho Integration Repository

pentaho

DI Repository

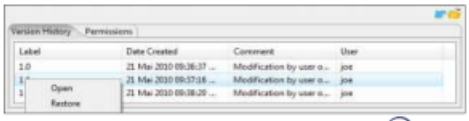
Enterprise Repository and Content Management

- Repository based on JCR (Content Repository API for Java)
- Repository browser
- Full revision history, allowing you to compare and restore previous revisions of a job or transformation
- Ability to lock transformations/jobs for editing
- 'Recycle bin' concept for working with deleted files

Enterprise security

- Configurable authentication including support for LDAP and MSAD
- Task permissions to control what actions a user/role can perform such as read/execute content, create content and administer security.

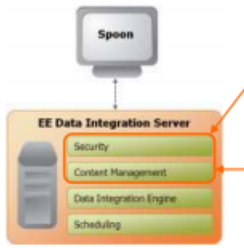
Scheduling



pentaho

Data Integration Repository

- Based on the security and content management modules in the EE Data Integration Server:



Security - Allows you to manage users and roles (default security) or integrate security into your existing security provider (such as LDAP or Microsoft Active Directory)

Content management - Provides the ability to centrally store and manage ETL jobs and transformations (includes full revision history of content and features such as sharing and locking for collaborative development environments)

pentaho

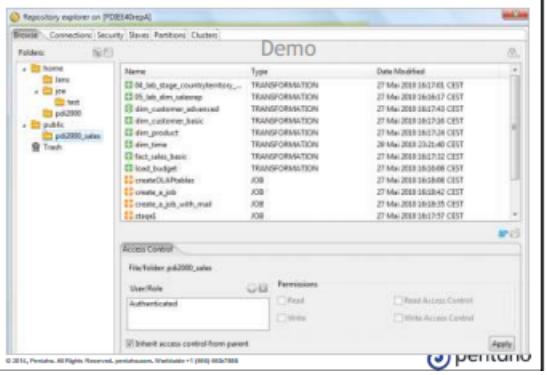
Connecting to the Repository When Files are Open

If you have files open when connecting to a repository, a message appears that varies depending on your permissions.

- *Would you like to close all open files?* This message appears if you have Create, Read, and Execute permissions. You can choose to close open transformation and job files or to leave them open. Click Yes or No.
- *You have limited permissions. Some functionality may not be available to you. Would you like to close all open files now?* This message appears if you have Read and Create permissions. You can choose to close open transformation and job files or to leave them open. Click Yes or No.

pentaho

Content Management



pentaho

Security

- The Data Integration Server is configured (by default) to use the Pentaho default security provider.
  - Pre-populated with a set of sample users and roles including:
    - Joe – Member of the administrator role with full access to and control over content in Data Integration Server
    - Suzy – Member of the CEO role with permission to read and create content, but without administrative access
  - Note: See the security guide in the Pentaho Documentation for details on configuring security with an existing security provider (such as LDAP or Microsoft Active Directory).

pentaho

Content Management

- Repository based on JCR (Content Repository API for Java)
- Enterprise security:
  - Configurable authentication including support for LDAP and MSAD
  - Task permissions defining what actions a user/role can perform such as read/execute content, create content, and administer security
  - Granular permissions on individual files and folders
- Full revision history on content allowing you to compare and restore previous versions of job or transformation
- Ability to lock transformations/jobs for editing
- Recycle bin concept for working with deleted files

pentaho

Exercise 11

Data Integration Repository

Time: 15 Minutes

Objectives:

- In this exercise you configure and use the Pentaho Enterprise Repository to perform common tasks.

Tasks:

- Configure the Pentaho Enterprise Repository, including basic security
- Create folders in the repository
- Add transformations and jobs to the repository
- Move, lock, and revise artifacts stored in the repository
- Delete and restore repository artifacts

pentaho

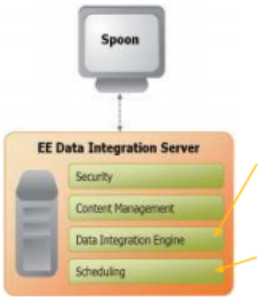
Module 9:

Scheduling & Monitoring

Lesson 1 – Setting up the Scheduler

pentaho

Scheduling in the DI Server



Data Integration Engine - This is a Carte instance. Carte is also used in clustering (covered later in this course).

Scheduling - The Quartz scheduler is used internally, and the tasks are executed in the data integration engine.

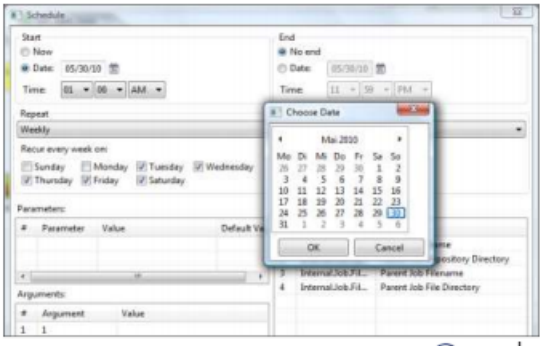
pentaho

Scheduling Options

- **Scheduling in PDI:**
  - Carte is integrated in the DI Server as the Data integration engine.
    - Scheduled jobs and transformations are executed in this Carte instance.
- Other scheduling options in PDI:
  - The operating system (CRON jobs, task scheduler)

pentaho

Exploring the Schedule Perspective (1 of 2)



pentaho

Content Management

- Repository based on JCR (Content Repository API for Java)
- Enterprise security:
  - Configurable authentication including support for LDAP and MSAD
  - Task permissions defining what actions a user/role can perform such as read/execute content, create content, and administer security
  - Granular permissions on individual files and folders
- Full revision history on content allowing you to compare and restore previous versions of job or transformation
- Ability to lock transformations/jobs for editing
- Recycle bin concept for working with deleted files

pentaho

Exercise 11

Data Integration Repository

Time: 15 Minutes

Objectives:

- In this exercise you configure and use the Pentaho Enterprise Repository to perform common tasks.

Tasks:

- Configure the Pentaho Enterprise Repository, including basic security
- Create folders in the repository
- Add transformations and jobs to the repository
- Move, lock, and revise artifacts stored in the repository
- Delete and restore repository artifacts

pentaho

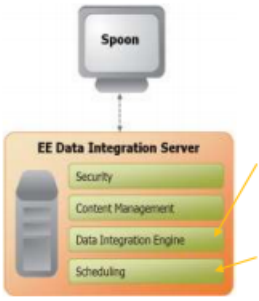
Module 9:

Scheduling & Monitoring

Lesson 1 – Setting up the Scheduler

pentaho

Scheduling in the DI Server



Data Integration Engine - This is a Carte instance. Carte is also used in clustering (covered later in this course).

Scheduling - The Quartz scheduler is used internally, and the tasks are executed in the data integration engine.

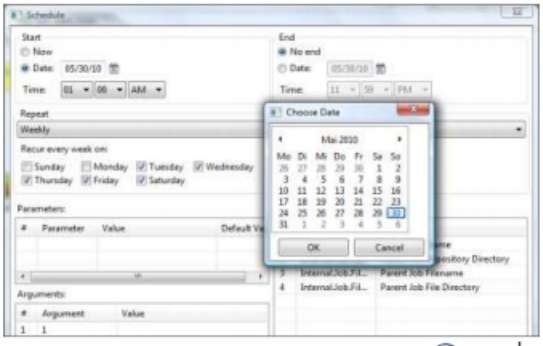
pentaho

Scheduling Options

- **Scheduling in PDI:**
  - Carte is integrated in the DI Server as the Data integration engine.
    - Scheduled jobs and transformations are executed in this Carte instance.
- Other scheduling options in PDI:
  - The operating system (CRON jobs, task scheduler)

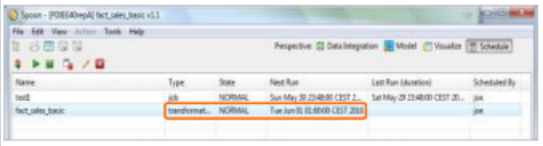
pentaho

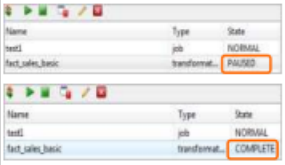
Exploring the Schedule Perspective (1 of 2)



pentaho

Exploring the Schedule Perspective (2 of 2)





- Note: Pause/complete does not mean the job or transformation is paused/completed. The option refers to scheduling.
- The Start and Stop buttons also refer to the scheduler (not the transformations or jobs).

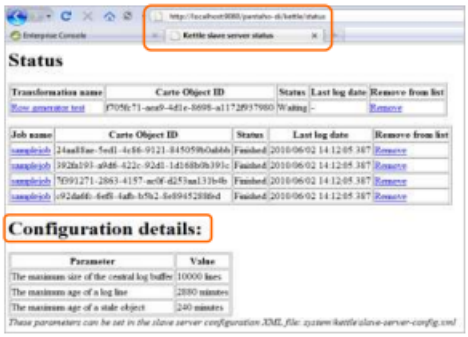
pentaho

Module 9:
Scheduling & Monitoring

Lesson 1 – Monitoring Scheduled Tasks

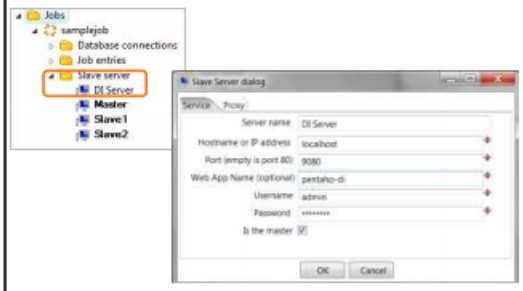
pentaho

Monitoring the DI Server: PDI Status



pentaho

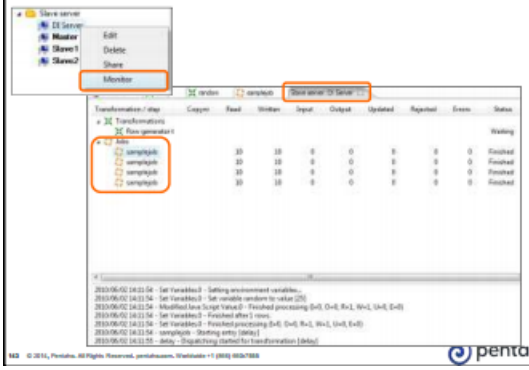
Monitoring in Spoon: Slave Server Status (1 of 2)



- Note: When monitoring the DI Server, you must enter the Web App Name: pentaho-di.

pentaho

Monitoring in Spoon: Slave Server Status (2 of 2)



pentaho

Guided Demo 11
Scheduling & Monitoring

Time: 20 Minutes

Objectives:

- In this exercise you configure job scheduling and monitoring using the Pentaho platform.

Tasks:

- Schedule the execution of a transformation using Pentaho Data Integration
- Monitor job execution on the Pentaho Data Integration server
- Using the Pentaho Enterprise Console, register a Carte server and transformation for monitoring
- View transformation step metrics in Carte
- View performance trend data in Carte

pentaho

Module 10:
Detailed Logging

pentaho

Logging

Logs contain summary information about job or transformation execution.

- Number of records inserted
- Total elapsed time spent in transformation

Logs also contain detailed information about job, transformation execution.

- Exceptions
- Errors
- Debugging information

pentaho

Reasons to Enable Logging

Reliability

- Job errors
- Review errors encountered

Headless operation

- Most ETL in production is not run from user interface
- Need location to view job results

Performance monitoring

- Useful information for current performance problems and capacity planning

pentaho

Two Types of Logging Can be Used

Log entries

- Traditional "logging"
- File-based approach
- Verbose

Database logging

- Summarized results
- RDBMS-based approach
- Concise and structured

NOTE: Both types can be used at the same time.

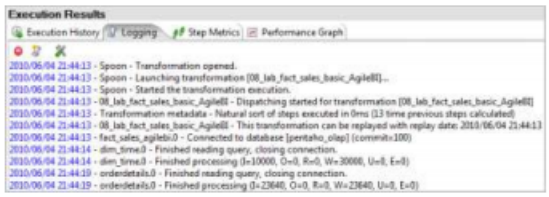
pentaho

Log Entries: Introduction

ALWAYS contains timestamp

Usually contains name of the step that logged the entry

Remainder varies according to the log entry



pentaho

Log Entries: File Locations

Default is `Spoon_xxx.log` in the temporary files folder, for example:
`\Users\Username\AppData\Local\temp`

spoon_a3579d51-96d9-11dc-a694-bf7a2db6f4d1.log	19.11.2007 20:58	LOG-Datei	1 KB
spoon_a8972bd2-9dab-11dc-911a-ad1edb33cf66.log	28.11.2007 13:15	LOG-Datei	0 KB
spoon_a945368e-9c0f-11dc-887e-fb7882a166e0.log	26.11.2007 13:28	LOG-Datei	125 KB
spoon_abba11d0-a117-11dc-a5a3-45a53dfa2b50.log	02.12.2007 21:49	LOG-Datei	1 KB
spoon_abceaef0-734c-11dc-a746-139673cc94fe.log	05.10.2007 15:51	LOG-Datei	16 KB
spoon_ac5a9f9d-8171-11dc-9310-6bb5aba34b13.log	23.10.2007 16:57	LOG-Datei	47 KB
spoon_adea125c-705d-11dc-84eb-6d97dfe019f1.log	01.10.2007 21:36	LOG-Datei	4 KB

Location can be set using command line:

- `kitchen.sh -logfile=/tmp/mylogfile.log`

Warning:

- Log files can become very large (several hundred MB), depending on logging levels
- Place logs in separate directory and periodically archive or purge

pentaho

Log Entries: Logging Levels

Level sets verbosity and information logged

Logging levels are additive:

- Basic = **Minimal + basic log entries**
- All the entries from the previous levels PLUS the level selected

Levels:

- Nothing
- Error
- Minimal
- Basic
- Detailed
- Debug
- Row level

Log Entries: Error and Minimal

Error:

- Builds on nothing
- Writes to log if errors occur
- If no errors occur, no log output

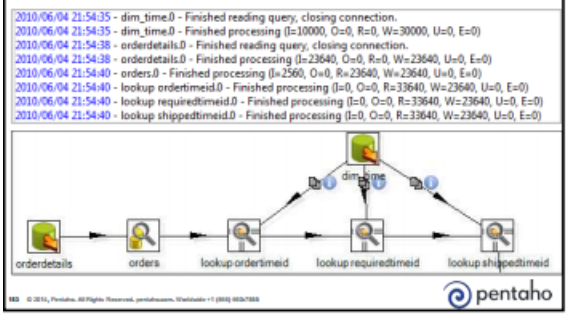
Minimal:

- Builds on Error
- Writes minimal log entries
- Typically transformation started and transformation ended

Log Entries: Basic

Basic:

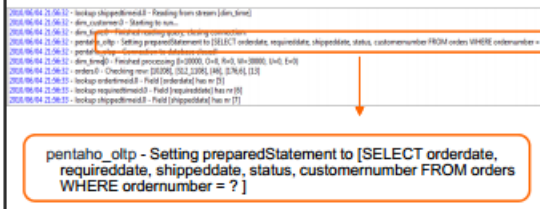
- Builds on Minimal
- Logs information on individual steps



Log Entries: Detailed

Detailed:

- Builds on Basic
- Each step provides MORE information about execution
  - Previous log level only writes summary for each step, Detailed writes additional step information
- Database steps provide information about database connection and statements executed



Log Entries: Debug and Row Level

Debug:

- Builds on Detailed
- VERBOSE: Logs nearly all information
- Useful for developers or for resolving obscure issues

Row Level:

- Builds on Debug
- MOST VERBOSE: Dumps row values as they pas through operators
- Useful when data value causes OraException with no helpful error message

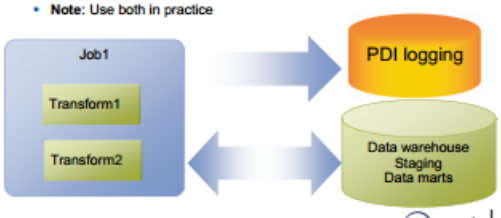
Database Logging: Introduction

Logs information into database tables in structured format for reporting and monitoring

Note: Connection to PDI logging and data warehouse can be same or different

Both jobs and transformations generate log entries

- Jobs log to table: pdi_log_job
- Transformations log to table: pdi_log_trans
- **Note**: Use both in practice



Database Logging and Kettle.properties

Several database logging definitions can be set by system variables.
There is no need to define this in every job and transformation.
The list of variables and descriptions can be viewed using the menu options – **Edit > Edit the Kettle.properties file**:

pentaho

Guided Demo 12

Detailed Logging throughout Execution

Time: 20 Minutes

Objectives:

- In this exercise you configure logging for job entries and for transformation steps.

Tasks:

- Create database tables to hold logging information from jobs and transformations
- Configure logging for transformation steps and for job entries
- Examine logging information produced by jobs and transformations

pentaho

Congratulations for your success in completing all course modules!

You have already learned a great deal since beginning this course a few days ago.

The remaining slides provide resource links where you can advance your PDI knowledge even more!

pentaho

Resources

- Kettle project page:
  http://kettle.pentaho.com
- Enterprise Edition Documentation
  http://help.pentaho.com/
- For up to date information, check the forums:
  http://forums.pentaho.org/forumdisplay.php?f=69
- Bug and Feature Requests with Road Maps (JIRA):
  http://jira.pentaho.com
- FAQ for Bug and Feature Requests:
  http://wiki.pentaho.com/display/EAI/Bug+Reports+and+Feature+Requests+FAQ

Join the conversation. You can find us on:

pentaho

pentaho

Please take a moment to provide us your feedback.

Please check now for an email sent to the account you provided when you registered for this class.

Within two minutes, you should have a survey. Filling this out now will help us consistently improve the training we provide to you.

Your feedback is greatly appreciated!

161