****

**Peer Teaching Learning Activity - TALEND ASSIGNMENT**

**File Management Manual**

**BY**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  | | |
| SL no. | Group Member Name | | |
| 1. | Sanket Dehankar | | |
| 2. | Rushi Kothari | | |
| 3. | Niranjan Patil | | |
| 4. | Mehul Patil | | |

**Guided By**

Mrs. Vaishali Wangikar

Mrs. Subhangi Kale

s

1. **How to Read Data from a Delimited File?**

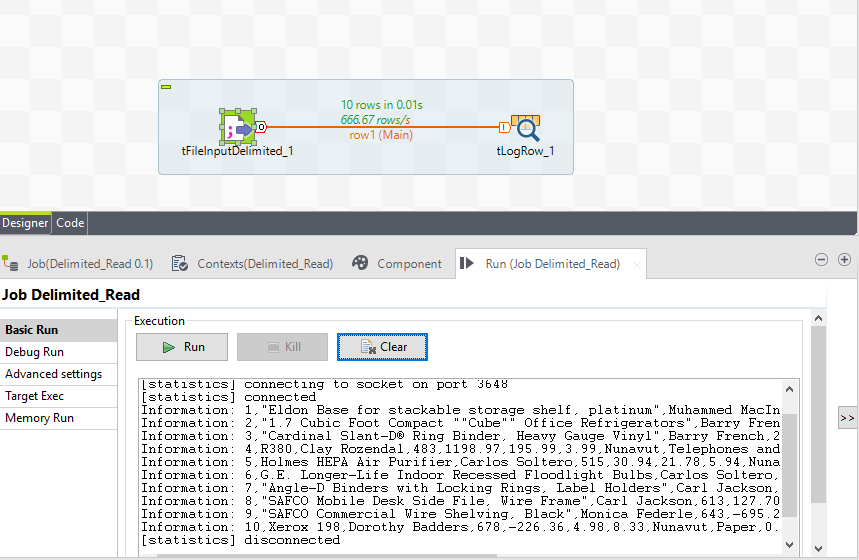
**Components Required:**

* tFileInputDelimited
* tLogRow

**Procedure:**

1. Drop a tFileInputDelimited component and a tLogRow component from the Palette to the design workspace.
2. Right-click on the tFileInputDelimited component and select Row > Main. Then drag it onto the tLogRow component and release when the plug symbol shows up.
3. Select the tFileInputDelimited component again, and define its Basic settings.
4. Define the Row separator allowing to identify the end of a row. Then define the Field separator used to delimit fields in a row.
5. Browse the csv file to be read and set its path. Further define header, footer as per required.
6. Set the Schema as either a local (Built-in) or a remotely managed (Repository) to define the data to pass on to the tLogRow component.
7. You can load and/or edit the schema via the Edit Schema function.
8. Select the tLogRow and define the Field separator to use for the output display.
9. Select the Print schema column name in front of each value check box to retrieve the column labels in the output displayed.
10. Press Ctrl+S to save your Job.
11. Go to Run tab, and click on Run to execute the Job.

**OUTPUT:**



1. **How to Read Data from an Excel File?**

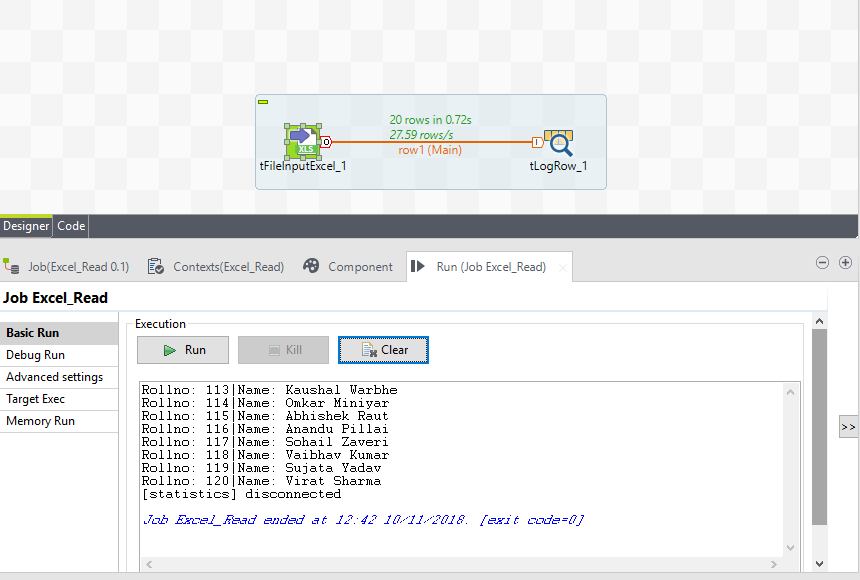
**Components Required:**

* tFileInputExcel
* tLogRow

**Procedure:**

1. Drop a tFileInputExcel component and a tLogRow component from the Palette to the design workspace.
2. Right-click on the tFileInputExcel component and select Row > Main. Then drag it onto the tLogRow component and release when the plug symbol shows up.
3. Select the tFileInputExcel component again, and define its Basic settings.
4. Browse the excel file to be read and set its path. Further define header, footer as per required.
5. Set the Schema as either a local (Built-in) or a remotely managed (Repository) to define the data to pass on to the tLogRow component.
6. You can load and/or edit the schema via the Edit Schema function.
7. Select the tLogRow and define the Field separator to use for the output display.
8. Select the Print schema column name in front of each value check box to retrieve the column labels in the output displayed.
9. Press Ctrl+S to save your Job.
10. Go to Run tab, and click on Run to execute the Job.

**OUTPUT:**



1. **How to Read Data Complete Row as a Column from a File?**

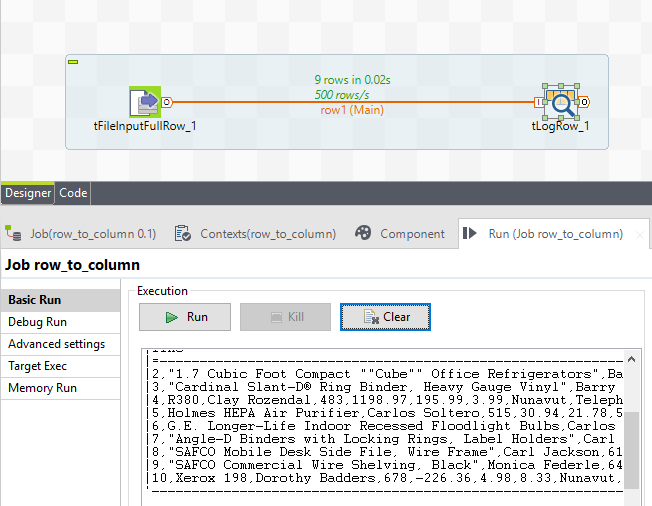
**Components Required:**

* tFileInputRow
* tLogRow

**Procedure:**

1. Create a new Job and add a tFileInputFullRow component and a tLogRow component by typing their names in the design workspace or dropping them from the Palette.
2. Link the tFileInputFullRow component to the tLogRow component using a Row > Main connection.
3. Double-click the tFileInputFullRow component to open its Basic settings view on the Component tab.
4. Click the [...] button next to Edit schema to view the data to be passed onto the tLogRow component. Note that the schema is read-only and it consists of only one column line.
5. In the File Name field, browse to or enter the path to the file to be processed.
6. In the Row Separator field, enter the separator used to identify the end of a row.
7. Double-click the tLogRow component to open its Basic settings view on the Component tab. In the Mode area, select Table (print values in cells of a table) for better readability of the result.
8. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT**



1. **How to write data to a Delimited File?**

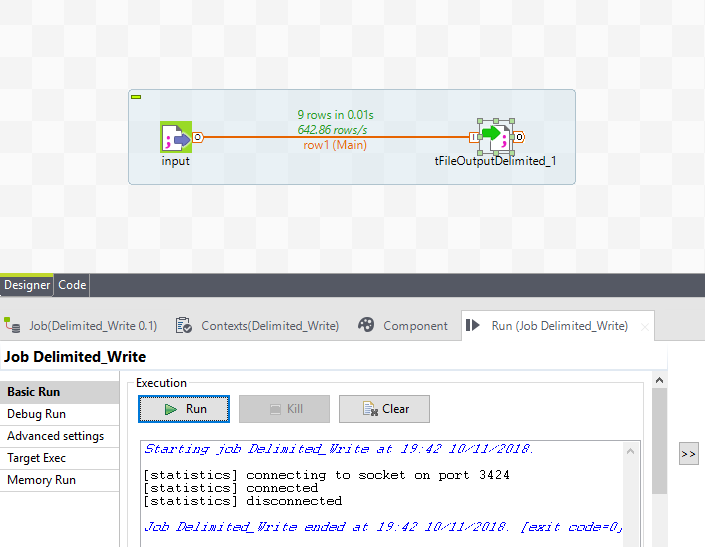
**Components Required:**

* tFileOutputDelimited
* tFileInputExcel or tFileInputDelimited

**Procedure:**

1. Drop a tFileInputExcel or tFileInputDelimited component and a tFileOutputDelimited component from the Palette onto the design workspace.
2. Right-click on tFileInputDelimited or tFileInputExcel and connect it to tFileOutputExcel using a Main Row link.
3. Click the Component tab to define the basic settings of tFileInputExcel or tFileInputDelimited. Browse the File name & other specify other details such as row separator, field separator etc.
4. Click the Component Tab of tFileOutputDelimited & Browse the File name & other specify other details such as append existing file, include header etc.
5. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT:**



1. **How to write data to an Excel File?**

**Components Required:**

* tFileOutputExcel
* tFileInputExcel or tFileInputDelimited

**Procedure:**

1. Drop a tFileInputExcel or tFileInputDelimited component and a tFileOutputExcel component from the Palette onto the design workspace.

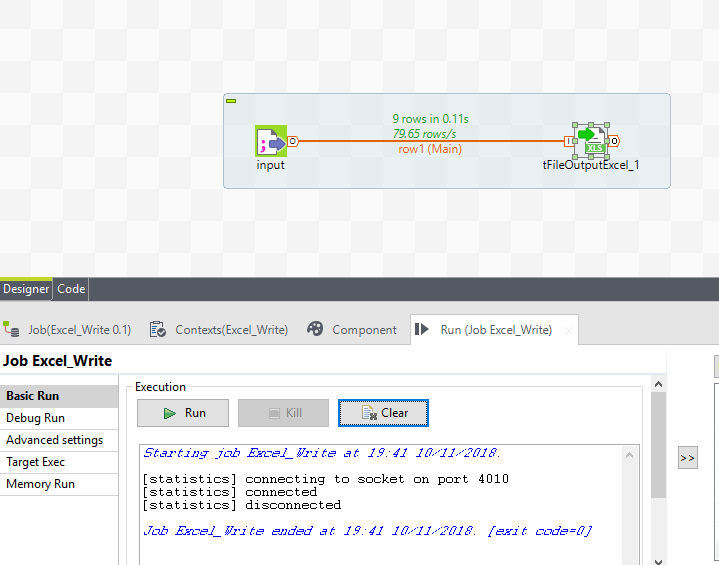
2. Right-click on tFileInputDelimited or tFileInputExcel and connect it to tFileOutputExcel using a Main Row link.

3. Click the Component tab to define the basic settings of tFileInputExcel or tFileInputDelimited. Browse the File name & other specify other details such as row separator, field separator etc.

4. Click the Component Tab of tFileOutputExcel & Browse the File name & other specify other details such as append existing file, include header etc.

5. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT:**



1. **How to find properties of a File?**

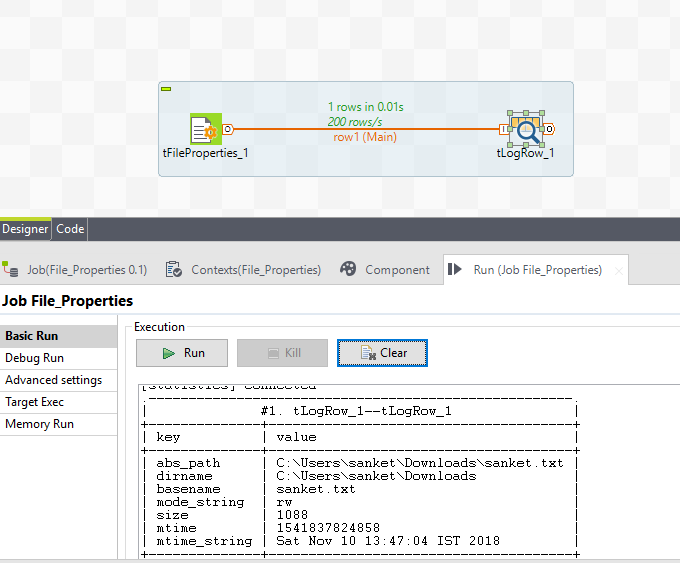
**Components Required:**

* tFileProperties
* tLogRow

**Procedure:**

1. Drop a tFileProperties component and a tLogRow component from the Palette onto the design workspace.
2. Right-click on tFileProperties and connect it to tLogRow using a Main Row link.
3. In the design workspace, select tFileProperties.
4. Click the Component tab to define the basic settings of tFileProperties.
5. Set Schema type to Built-In.
6. In the File field, enter the file path or browse to the file you want to display the properties for.
7. In the design workspace, select tLogRow and click the Component tab to define its basic settings. Press Run to execute the Job.

**OUTPUT**



1. **How to Unarchive or Unzip a File?**

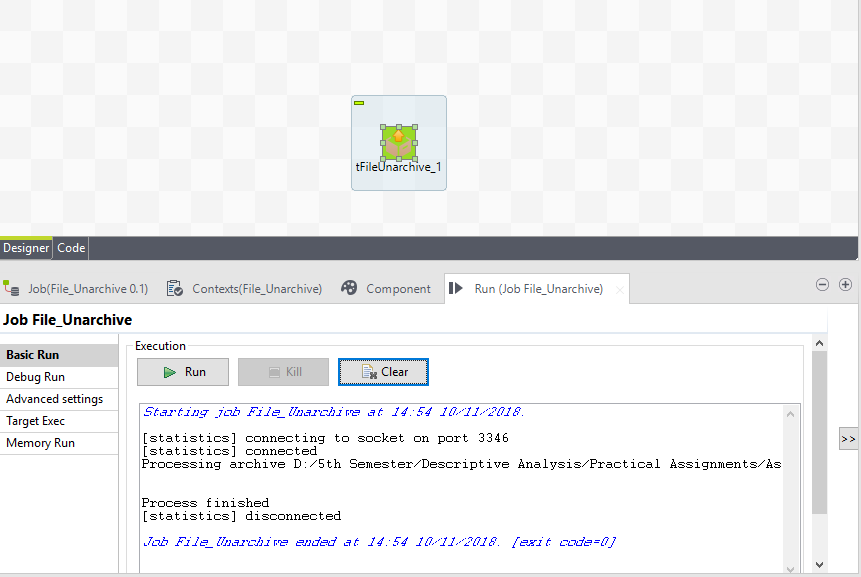
**Components Required:**

* tFileUnarchive

**Procedure:**

1. Drop a tFileUnarchive component from the Palette onto the design workspace.
2. Click the Component tab to define the basic settings of tFileUnarhive.
3. Click on Archive File to set the archived file path.
4. Select Extraction directory to enter the folder where the zip files are to be archived.
5. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT:**



1. **How to count the Rows in a File?**

**Components Required:**

* tFileRowCount
* tjava

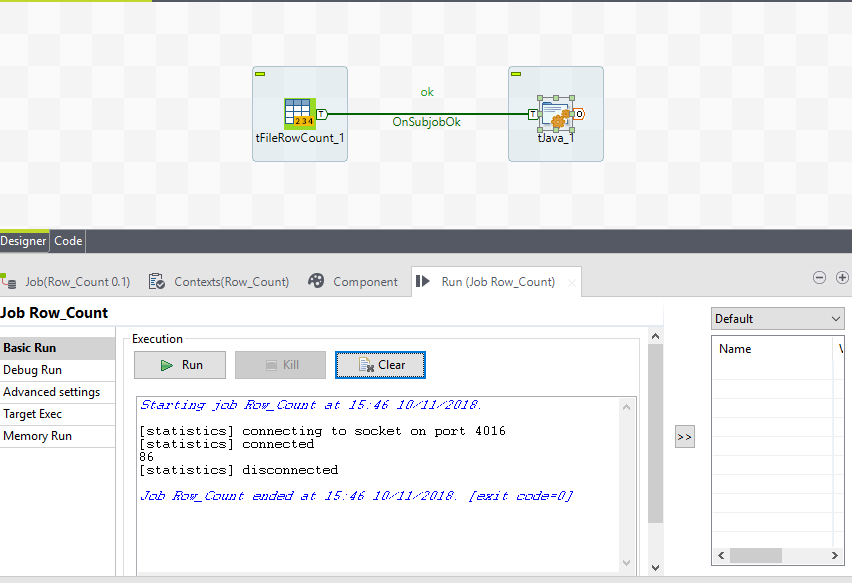
**Procedure:**

1. Drop a tFileRowCount component and a tjava component from the Palette onto the design workspace.
2. Right-click on tFileRowCount and connect it to tjava using a OnSubJobOk link.
3. Click the Component tab to define the basic settings of tFileRowCount.
4. In the File field, enter the file path or browse to the file you want to display the properties for. Select the Row Separator as per the requirement.
5. Click on the tJava component to write the following code:

System.out.println(globalMap.get("tFileRowCount\_1\_COUNT"));

1. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT:**



1. **How to List files in a Directory?**

**Components Required:**

* tFileList
* tIterateToFlow
* tLogRow

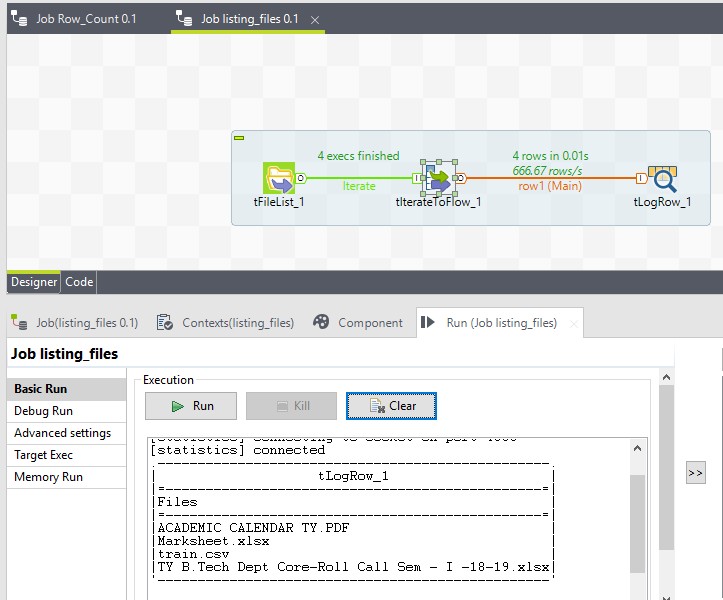
**Procedure:**

1. Drop a tFileList, tIterateToFlow & tLogRow components from the Palette onto the design workspace.
2. Right-click on tFileList component and connect it to tIterateToFlow using an iterate link & connect tIterateToFlow to tLogRow using a main link.
3. Click the Component tab to define the basic settings of tFileList.
4. In the Basic settings view and from the FileList Type list, select the source type you want to process, Files in this example.
5. In the Case sensitive list, select a case mode, Yes in this example to create case sensitive filter on file names.
6. Keep the Use Glob Expressions as Filemask check box selected if you want to use global expressions to filter files, and define a file mask in the Filemask field.
7. In the Basic Settings of tIterateToFlow component select a column from the schema & write the following code:

((String)globalMap.get("tFileList\_1\_CURRENT\_FILE"))

1. Specify the basic setting of tLogRow as previously done.
2. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT:**



1. **How to Delete a File?**

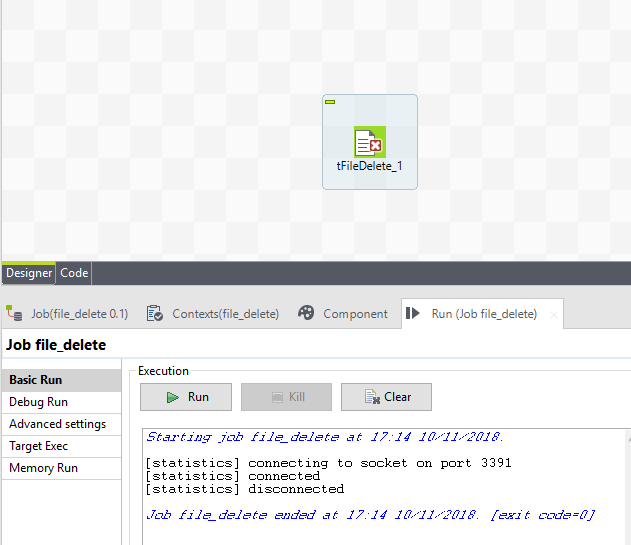
**Components Required:**

* tFileDelete

**Procedure**

1. Drop a tFileDelete component from the Palette onto the design workspace.
2. Click the Component tab to define the basic settings of tFileDelete.
3. Select the Directory or path to be deleted as per required.
4. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT:**



1. **How to Copy Files?**

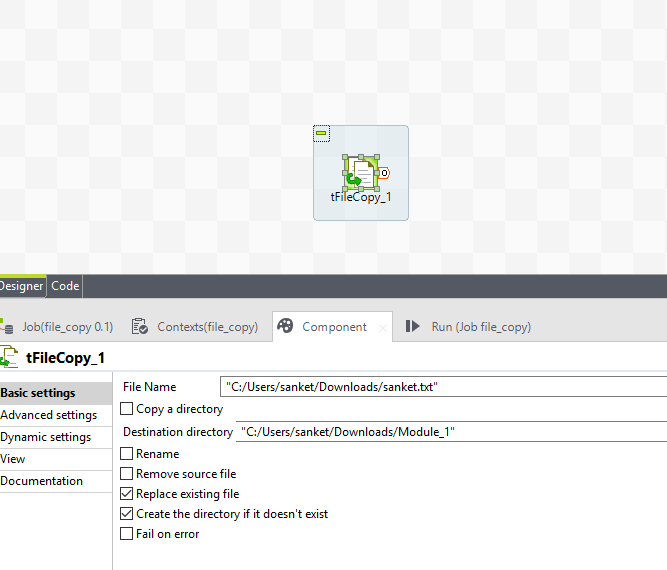
**Components Required:**

* TFileCopy

**Procedure:**

1. Drop a tFileCopy component from the Palette onto the design workspace.
2. Click the Component tab to define the basic settings of tFileCopy.
3. Select the File Name to be copied and the Destination Directory as per required.
4. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT:**



1. **How to Compare Two Files?**

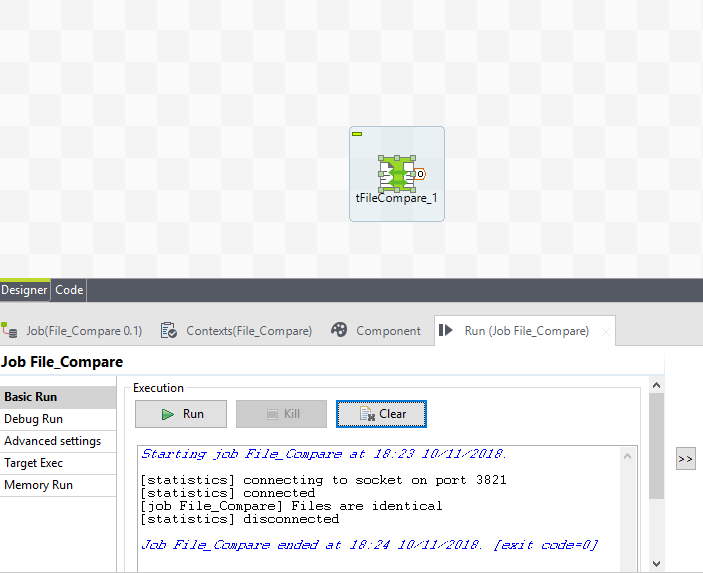
**Components Required:**

* TFileCompare

**Procedure:**

1. Drop a tFileCompare component from the Palette onto the design workspace.
2. Click the Component tab to define the basic settings of tFileCompare.
3. Browse the File to Compare & further browse the Reference File to be processed for comparing.
4. Specify the message to be displayed on console whether files differ or are identical.
5. Press Ctrl+S to save your Job and then press Run to execute the Job.

**OUTPUT:**



1. **How to Create a Temporary File?**

**Components Required:**

* tCreateTemporaryFile
* tJava
* tRowGenerator
* tFileOutputDelimited
* tFileInputDelimited
* tLogRow .

**Procedure:**

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a tCreateTemporaryFile component, a tJava component, a tRowGenerator component, a tFileOutputDelimited component, a tFileInputDelimitedcomponent, and a tLogRow component.
2. Connect tRowGenerator to tFileOutputDelimited using a Row > Main connection.
3. Do the same to connect tFileInputDelimited to tLogRow.
4. Connect tCreateTemporaryFile to tJava using a Trigger > OnSubjobOk connection.
5. Do the same to connect tJava to tRowGenerator and connect tRowGenerator to tFileInputDelimited.

Configuring the components

Creating the temporary file

1. Double-click tCreateTemporaryFile to open its Basic settings view.
2. Select the Remove file when execution is over check box to delete the created temporary file after the Job execution.
3. Select the Use default temporary system directory check box to create the file in the default system temporary directory.
4. In the Template field, enter the temporary file name which should contain the characters *XXXX*. In this example, it is *talend\_XXXX*.
5. In the Suffix field, enter the filename extension of the temporary file. In this example, it is *dat*.

What is a DAT file?

A DAT file is a generic data file created by a specific application. It may contain data in binary or text format (text-based DAT files can be viewed in a text editor). DAT files are typically accessed only by the application that created them.

1. Double-click tJava to open its Basic settings view.
2. In the Code field, enter the following code to display the default system temporary directory and the path to the temporary file that will be created on the console:
3. System.out.println("The default system temporary directory is:\r" + (String)System.getProperty("java.io.tmpdir"));

System.out.println("The path to the temporary file is:\r" + (String)globalMap.get("tCreateTemporaryFile\_1\_FILEPATH"));

Writing the data into the file

1. Double-click tRowGenerator to open its RowGenerator Editor.
2. Click the [+] button to add two columns: *id* of Integer type and *name* of String type. Then in the Functions column, select the predefined functionNumeric.sequence(String,int,int) for *id* and TalendDataGenerator.getFirstName() for *name*.
3. In the Number of Rows for RowGenerator field, enter *5* to generate five rows.
4. Click OK to validate the changes and accept the propagation prompted by the pop-up dialog box.
5. Double-click tFileOutputDelimited to open its Basic settings view.
6. In the File Name field, press Ctrl+Space and from the global variable list displayed select *((String)globalMap.get("tCreateTemporaryFile\_1\_FILEPATH"))*.

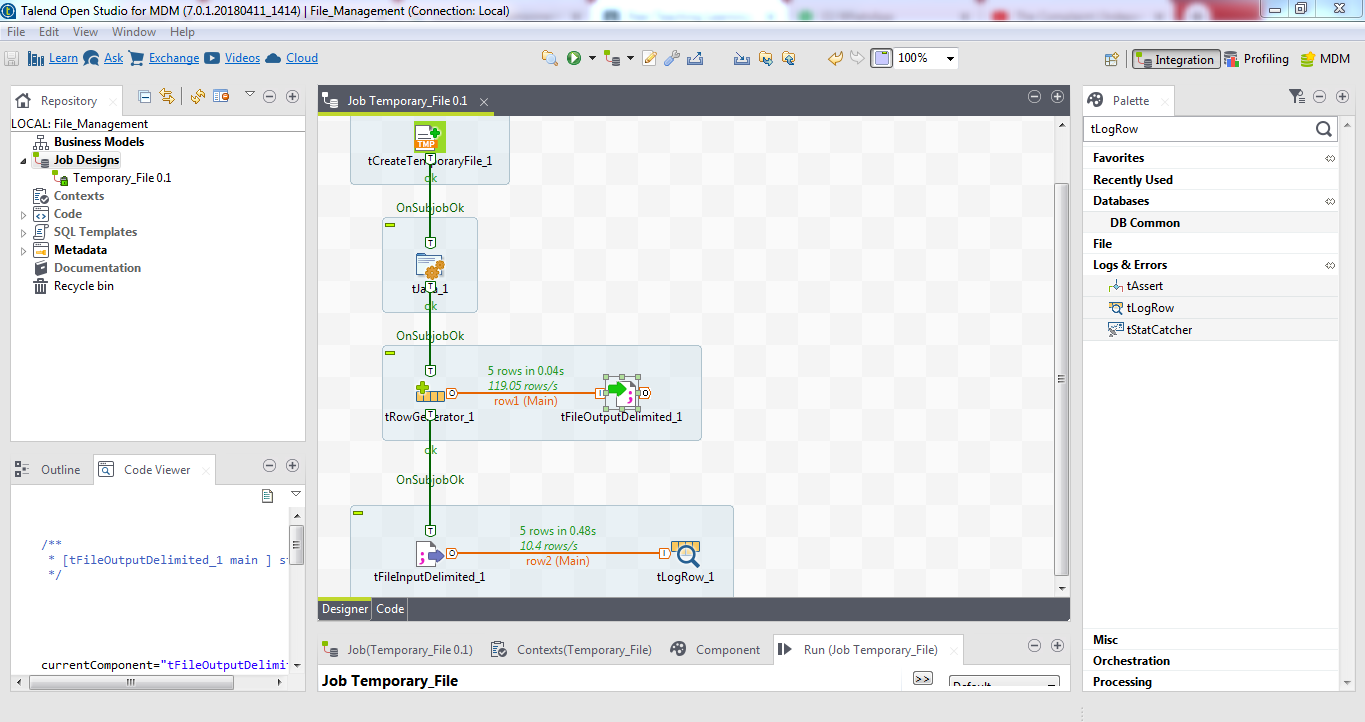
Reading the data from the file

1. Double-click tFileInputDelimited to open its Basic settings view.
2. In the File name/Stream field, press Ctrl+Space and from the global variable list displayed select *((String)globalMap.get("tCreateTemporaryFile\_1\_FILEPATH"))*.
3. Click the [...] button next to Edit schema and in the dialog box displayed define the schema by adding two columns: *id* of Integer type and *name*of String type.
4. Click OK to validate the changes and accept the propagation prompted by the pop-up dialog box.
5. Double-click tLogRow to open its Basic settings view.
6. In the Mode area, select Table (print values in cells of a table) to display the output data in a better way.

Saving and executing the Job

1. Press Ctrl+S to save the Job.
2. Press F6 or click Run on the Run tab to run the Job.
3. The file *talend\_MHTI.dat* is created under the default system temporary directory *C:\Users\lena\_li\AppData\Local\Temp\* during the Job execution, the five generated rows of data is written into it, then the file is deleted after the Job execution.

**OUTPUT:**



1. **How to Change Encoding of a File?**

**Components Required:**

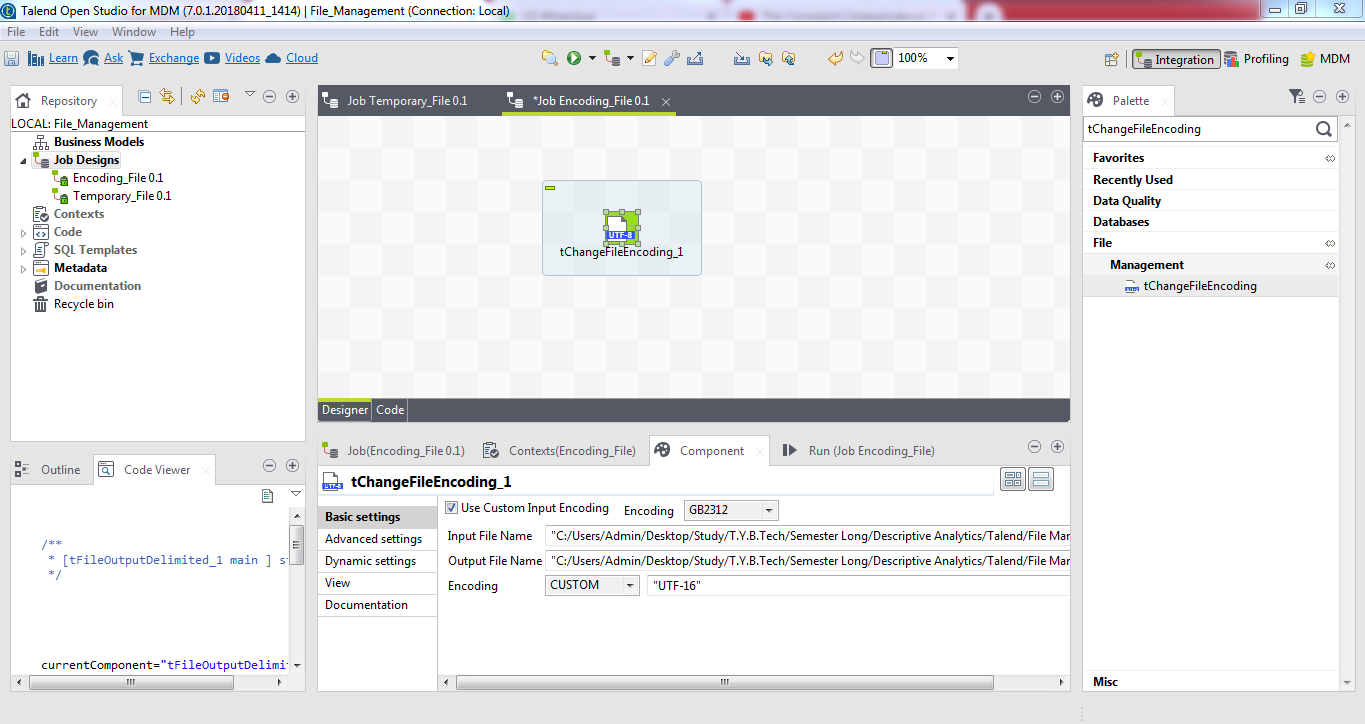
* tChangeFileEncoding

**Procedure:**

1. Drop a tChangeFileEncoding component onto the design workspace.
2. Double-click the tChangeFileEncoding component to display its Basic settings view.
3. Select Use Custom Input Encoding check box. Set the Encoding type to GB2312.
4. In the Input File Name field, enter the file path or browse to the input file.
5. In the Output File Name field, enter the file path or browse to the output file.
6. Select CUSTOM from the second Encoding list and enter *UTF-16* in the text field.
7. Press F6 to execute the Job.

**OUTPUT:**

The encoding type of the file *in.txt* is transformed and *out.txt* is generated with the UTF-16 encoding type.



1. **How to Archive a File?**

**Components Required:**

* tFileArchive

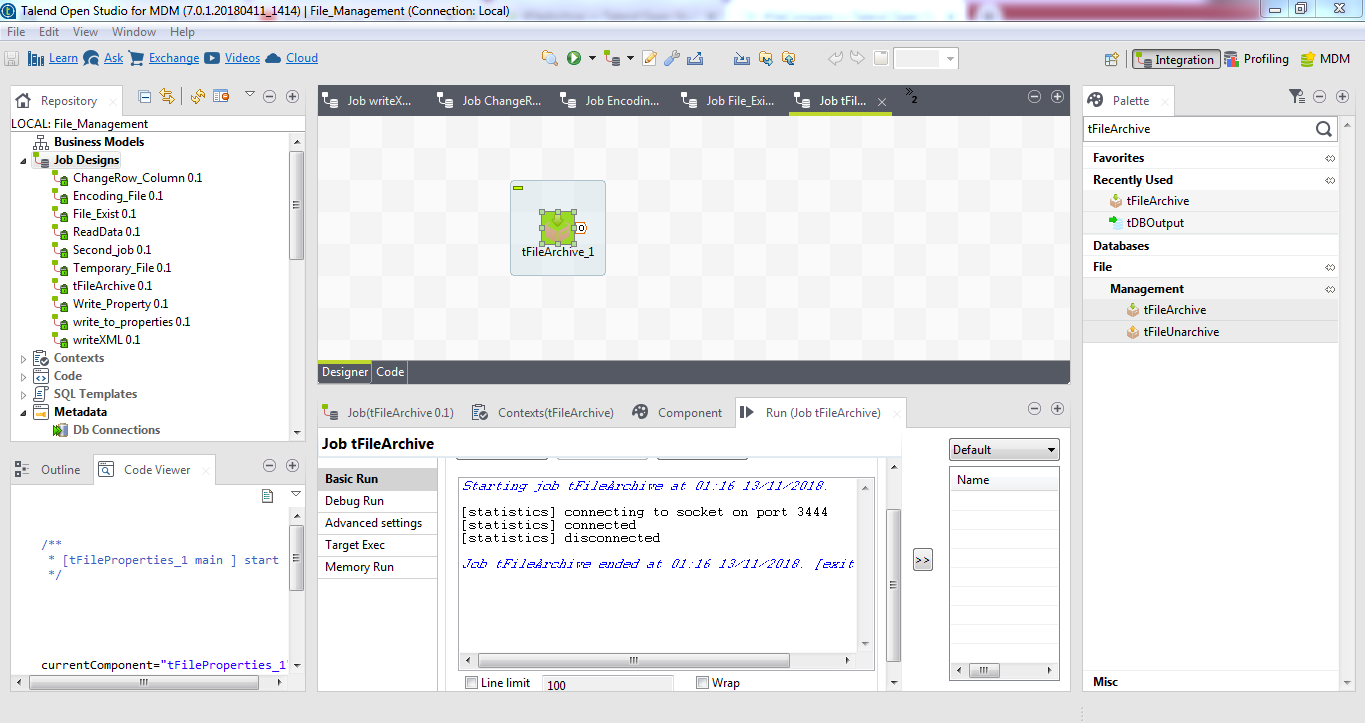
**Procedure:**

Scenario: Zip files using a tFileArchive

This scenario creates a Job with a unique component. It aims at zipping files and recording them in the selected directory.

1. Drop the tFileArchive component from the Palette onto the workspace.
2. Double-click it to display its Component view.
3. In the Directory field, click the [...] button, browse your directory and select the directory or the file you want to compress.
4. Select the Subdirectories check box if you want to include the subfolders and their files in the archive.
5. Then, set the Archive file field, by filling the destination path and the name of your archive file.
6. Select the Create directory if not exists check box if you do not have a destination directory yet and you want to create it.
7. In the Compress level list, select the compression level you want to apply to your archive. In this example, we use the normal level.
8. Clear the All Files check box if you only want to zip specific files.
9. Add a row in the table by clicking the [+] button and click the name which appears. Between two star symbols (ie. \*RG\*), type part of the name of the file that you want to compress.
10. Press F6 to execute your Job.

**OUTPUT:**

****

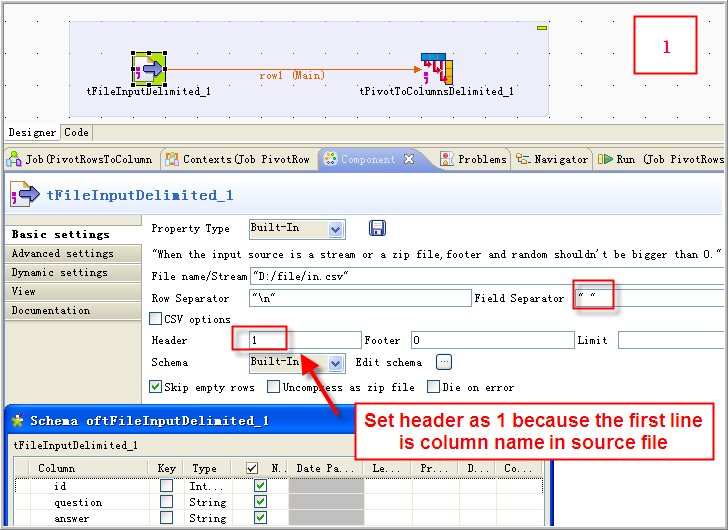
1. **How to Change Rows into Columns?**

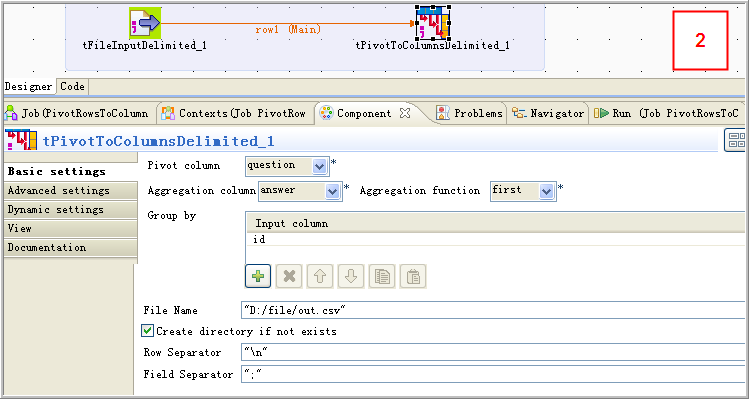
**Components Required:**

* **TPivotToColumnsDelimited**
* TFileInputDelimited

**Procedure:**

1. Create a simple Job as seen below. This job is also available as an attachment.





The **tPivotToColumnsDelimited**component is used to convert the rows to columns. It requires at least three columns in the input schema: the Pivot column, the Aggregation column, and one or more Group keys.

### 2.

Execute the Job. The output file will be as follows:

id;Name;Sex;Surgery

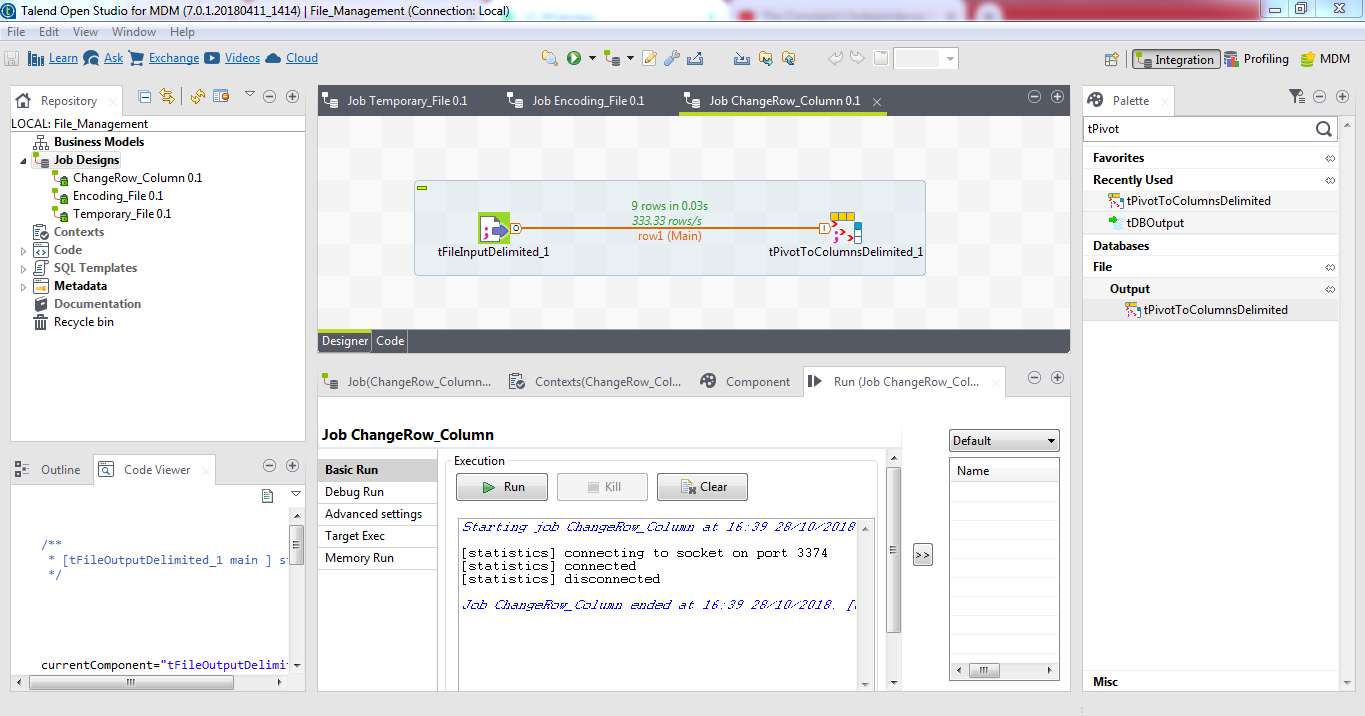
1;Shong;M;Yes

2;Ross;M;No

3;Elise;F;Yes

**Note**: It is not currently possible to output the result as a data flow that doesn't require an output file. If necessary, you can use a **tFileInputDelimited** component to read the result back into the next Job from the output file created by **tPivotToColumnsDelimited**.

**OUTPUT:**

****

1. **How to Check whether a File Exist or not?**

**Components Required:**

**Procedure:**

# Scenario: Checking for the presence of a file and creating it if it does not exist

This scenario describes a simple Job that: checks if a given file exists, displays a graphical message to confirm that the file does not exist, reads the input data in another given file and writes it in an output delimited file.

A dialog box appears to confirm that the file does not exists.

Click OK to close the dialog box and continue the Job execution process. The missing file, file1 in this scenario, got written in a delimited file in the defined place.

1. Drop the following components from the Palette onto the design workspace: tFileExist, tFileInputDelimited, tFileOutputDelimited, and tMsgBox.
2. Connect tFileExist to tFileInputDelimited using an OnSubjobOk and to tMsgBox using a Run If link.
3. Connect tFileInputDelimited to tFileOutputDelimite using a Row Main link.

# Configuring the components

1. In the design workspace, select tFileExist and click the Component tab to define its basic settings.
2. In the File name field, enter the file path or browse to the file you want to check if it exists or not.
3. In the design workspace, select tFileInputDelimited and click the Component tab to define its basic settings.
4. Browse to the input file you want to read to fill out the File Name field.

Warning:

If the path of the file contains some accented characters, you will get an error message when executing your Job.

1. Set the row and field separators in their corresponding fields.
2. Set the header, footer and number of processed rows as needed. In this scenario, there is one header in our table.
3. Set Schema to Built-in and click the Edit schema button to define the data to pass on to the tFileOutputDelimited component. Define the data present in the file to read, file2 in this scenario.

For more information about schema types, see *Talend Studio* User Guide.

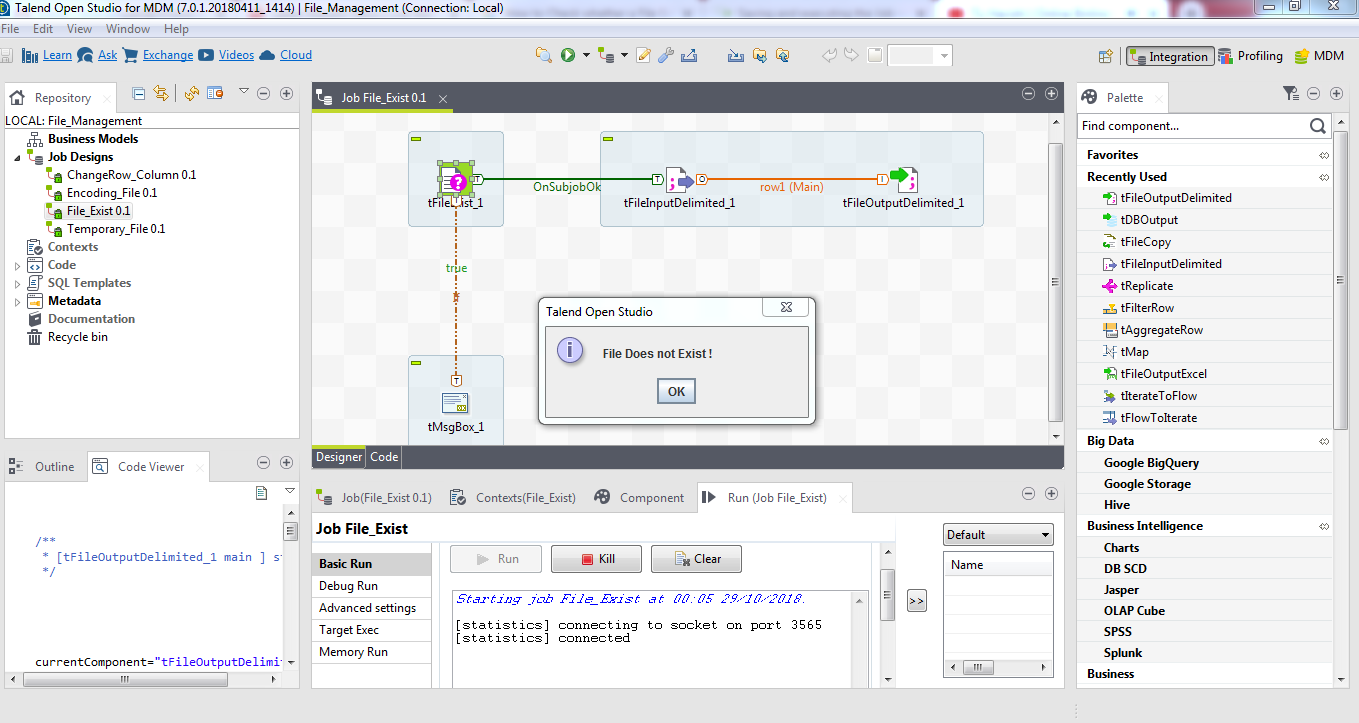
The schema in file2 consists of five columns: Num, Ref, Price, Quant, and tax.

1. In the design workspace, select the tFileOutputDelimited component.
2. Click the Component tab to define the basic settings of tFileOutputDelimited.
3. Set property type to Built-in.
4. In the File name field, press Ctrl+Space to access the variable list and select the global variable FILENAME.
5. Set the row and field separators in their corresponding fields.
6. Select the Include Header check box as file2 in this scenario includes a header.
7. Set Schema to Built-in and click Sync columns to synchronize the output file schema (file1) with the input file schema (file2).
8. In the design workspace, select the tMsgBox component.
9. Click the Component tab to define the basic settings of tMsgBox.
10. Click the If link to display its properties in the Basic settings view.
11. In the Condition panel, press Ctrl+Space to access the variable list and select the global variable EXISTS. Type an exclamation mark before the variable to negate the meaning of the variable.

# Saving and executing the Job

1. Press Ctrl+S to save your Job.
2. Press F6 or click the Run button in the Run tab to execute it.

**OUTPUT:**



1. **How to Write Data to a Properties File?**

**Components Required:**

* TFileProperties
* tFileOutputDelimited

**Procedure:**

1. Drop a tFileProperties component and a tFileOutputDelimited component rom the Palette onto the design workspace.

2.Right-clickon tFileProperties andconnectit to  tFileOutputDelimited using a Main Row link.

3.In the design workspace, select tFileProperties.

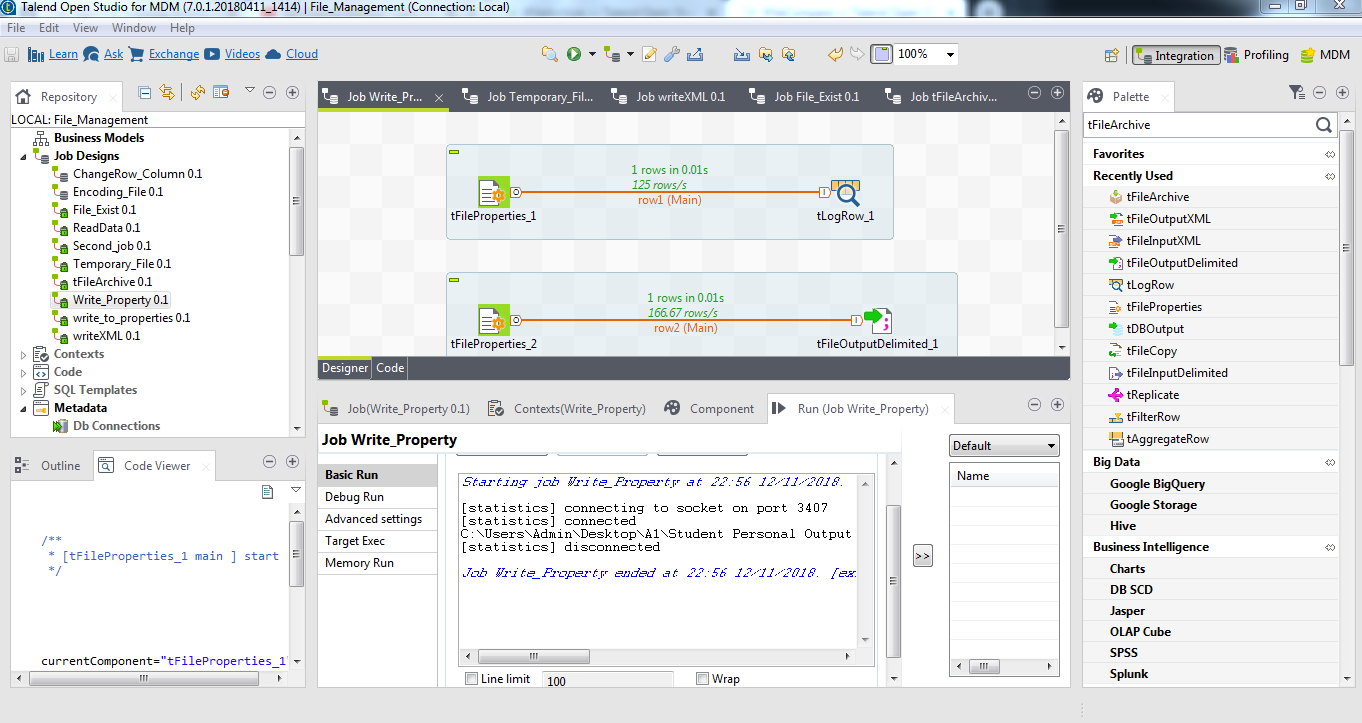
4. Click the Component tab to define the basic settings of tFileProperties.

5.Set Schema type to Built-In.

6.In the File field, enter the file path or browse to the file you want to WRITE the properties for.

7.In the design workspace, select tFileOutputDelimited and click the Component tab to define its basic settings. Press Run to execute the Job.

**OUTPUT:**

****

1. **How to Read Data From an Email File ?**

**Components Required:**

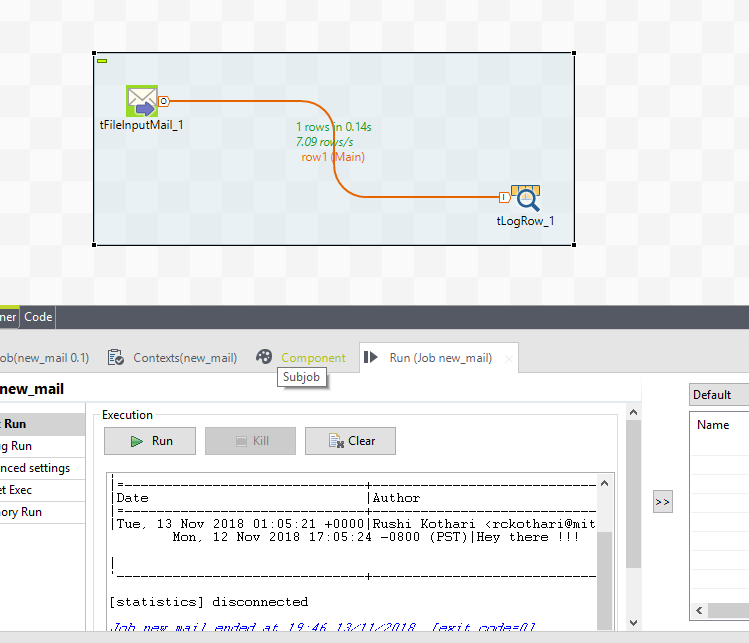
TfileInputEmail

TLogrow

**Procedure:**

* 1. Download one mail .file from Gmail file
  2. Then just edit the schema and add DATE,AUTHOR,OBJECT,STATUS,BODY to your schema.
  3. Also configure the main value in double quotes.
  4. Then connect it to a tlogrow and run it.

**OUTPUT:**

****

1. **How to Read Data From a Multi Structured Delimited File ?**

**Components Required:**

TfileinputMSDelimeted

Tlogrow

**Procedure:**

1. Double-click **tFileInputMSDelimited**to open the **Multi Schema Editor**.
2. Click **Browse...** next to the **File name** field to locate the multi schema delimited file you need to process.
3. In the **File Settings** area:

-Select from the list the encoding type the source file is encoded in. This setting is meant to ensure encoding consistency throughout all input and output files.

-Select the field and row separators used in the source file.

4. Once you have your file then you press the preview

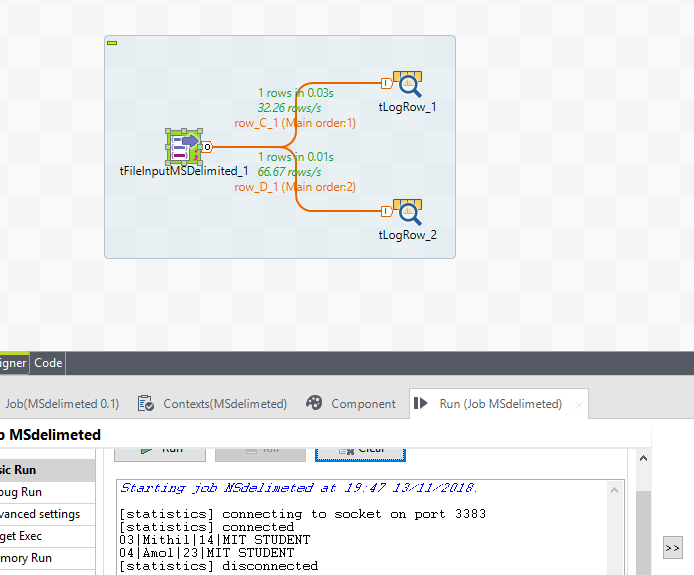
5. After that you press Fetch codes

6.After that you have to click on each row in section below fetch codes and click each row and then press the 🡪 symbol.

7. You have to do it for all fetch data

8. After that You just take TlogRow from pallete and then right click on MSdelimited and select row

And join it to the TlogRow

**Output:**

1. **How to Read Data From a Multi Structured Positional File?**

**Components Required:**

TfileinputMSpositional

tlogrow

**Procedure:**

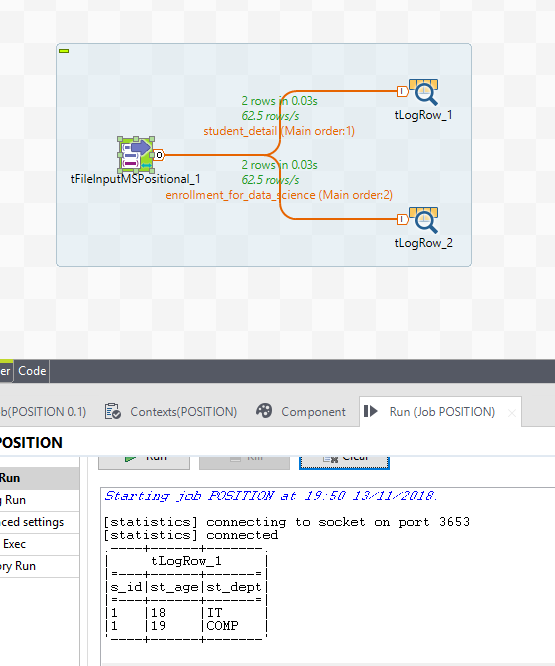
\*.Double-click the **tFileInputMSPositional** component to show its **Basic settings** view and define its properties

1. In the **File name/Stream** field, type in the path to the input file. Also, you can click the **[...]** button to browse and choose the file.
2. In the **Header Field Position** field, enter the start-end position for the schema identifier in the input file, *0-1* in this case as the first character in each row is the schema identifier.
3. Click the **[+]** button twice to added two rows in the **Records** table.
4. Click the cell under the **Schema** column to show the **[...]** button.

Click the **[...]** button to show the schema naming box.

1. Then add the schema name and in header value write the key you in your schema
2. In pattern you have to add length of the schema field
3. And in last you have ability to skip the header section then you can write the value from which you have to skip the header
4. Lastly join it to the tlog row and see the output

**OUTPUT:**

****

1. **How to Read Data From a XML File ?**

**Components Required:**

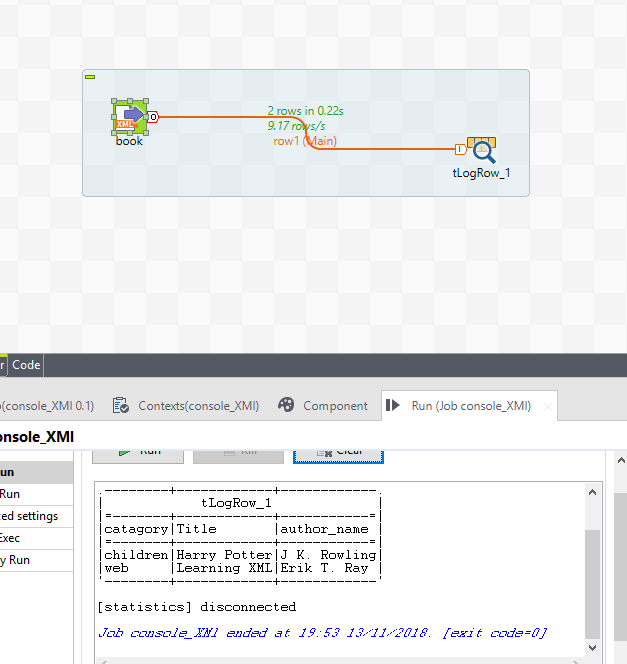
XML metadata

Tlogrow

**Procedure:**

* 1. From the repository just create one xml file metadata but in step 4 you have to give an absolute path and relative path of your XML file.
  2. And When you are done with metadata then add it to the workspace
  3. Then Double click on it
  4. Then by clicking … symbol just browse the XML file
  5. Then in localXpath you have to give the node path to which you have to acces the data.
  6. In Mapping section add the column name and in Xpath query field you have to give the tag name in XML file.
  7. Then Join it to the TlogRow and see the output.

**OUTPUT:**



1. **How to Read Data From a Multi Structured XML File ?**

**Components Required:**

* tfileInputMSxml
* togrow

**Procedure:**

1.Add the XML file

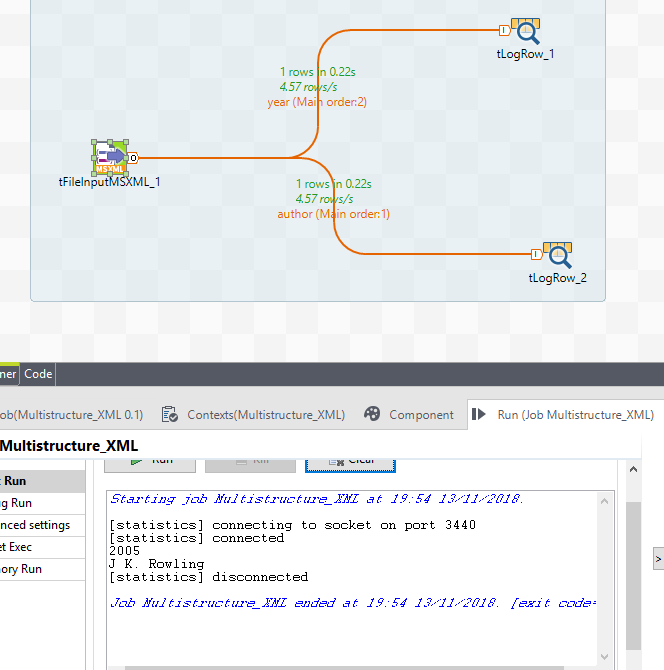
2, In Root query add the name of your root tag

3.In schema section write the tag names you want to access

4.In schema Xpathloop just add the / symbol to your tag names

5.In Xpath query add the . symbol on “”

6.Then join to the tlogRow

**OUTPUT:**

1. **How to Read Data from a Properties File?**

**Components Required:**

* TFileProperties
* tLogRow

**Procedure:**

1. Drop a tFileProperties component and a tLogRow component rom the Palette onto the design workspace.

2.Right-clickon tFileProperties andconnectit to  tLogRowusing  Main Row link.

3.In the design workspace, select tFileProperties.

4. Click the Component tab to define the basic settings of tFileProperties.

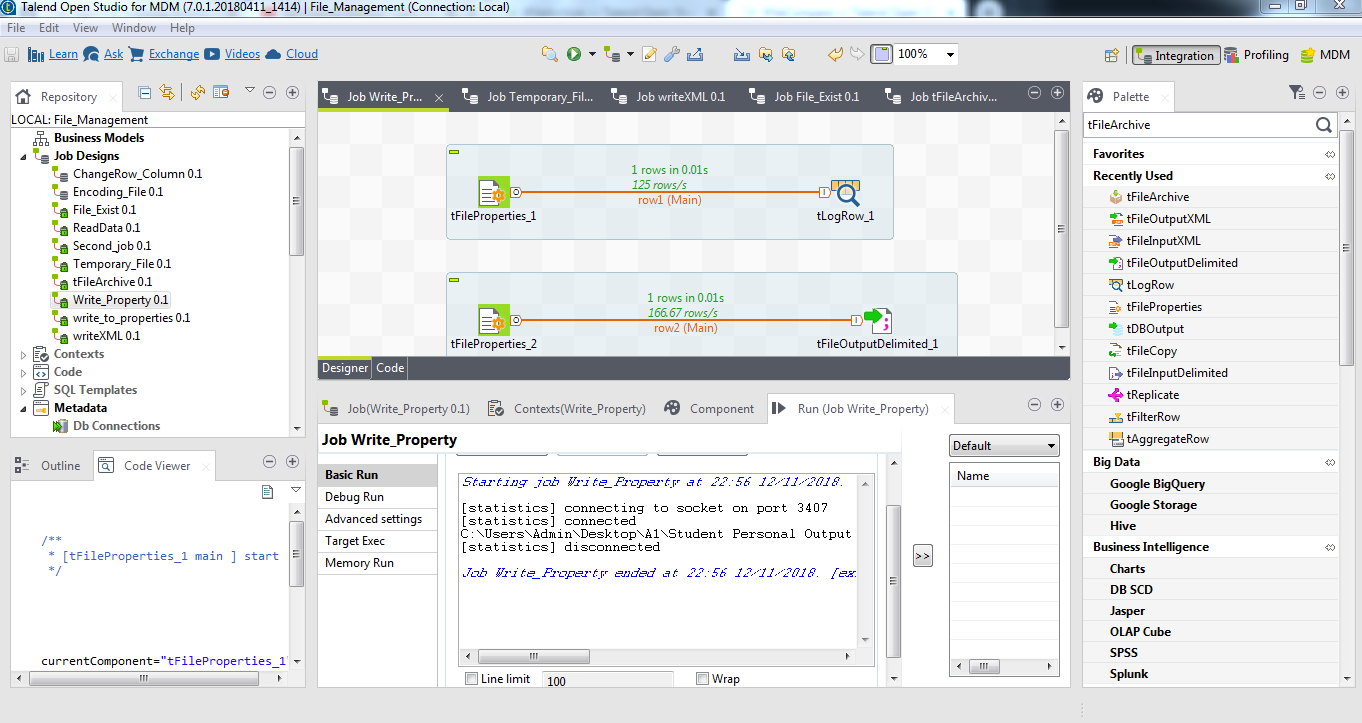
5.Set Schema type to Built-In.

6.In the File field, enter the file path or browse to the file you want to WRITE the properties for.

7.In the design workspace, select tLogRow

 and click the Component tab to define its basic settings. Press Run to execute the Job.

**OUTPUT:**

****

1. **How to Read all the content of a file and Send it to a Single column?**

**Components Required:**

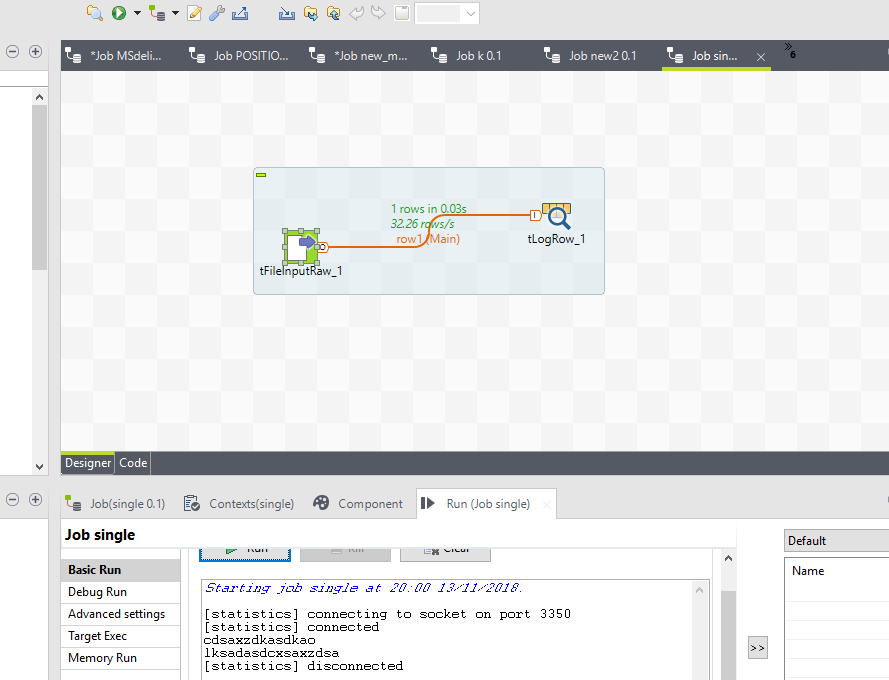
* TfileInputRow
* TlogRow

**Procedure:**

1.Double click on the TfileInput Row and add the file to it

2.Then Read the file as string

3. Then Join the file to Tlog row and display the output.

**OUTPUT: **

1. **How to Read Data From a Regex File?**

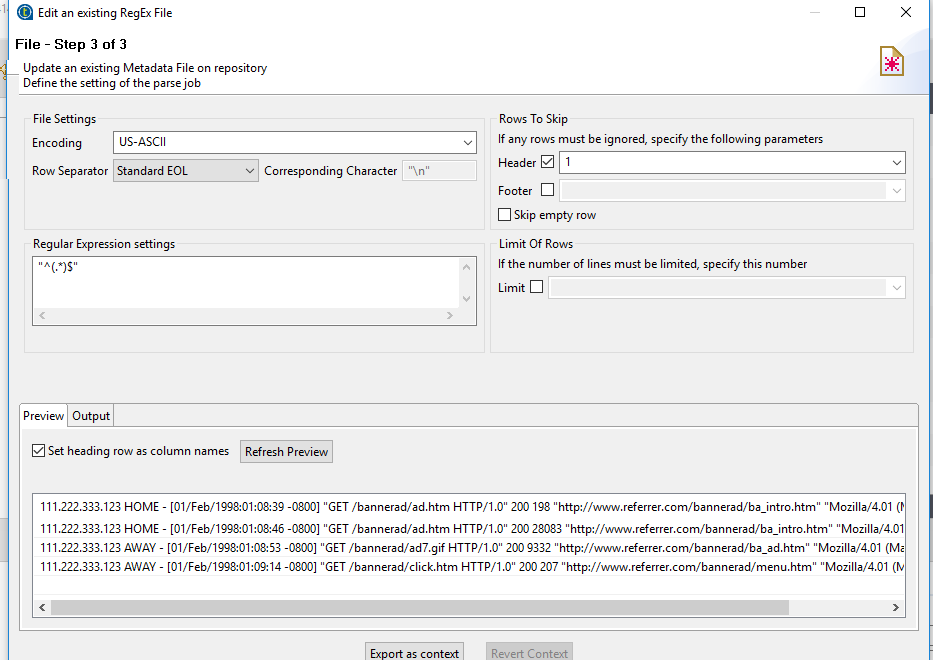
**Components Required:**

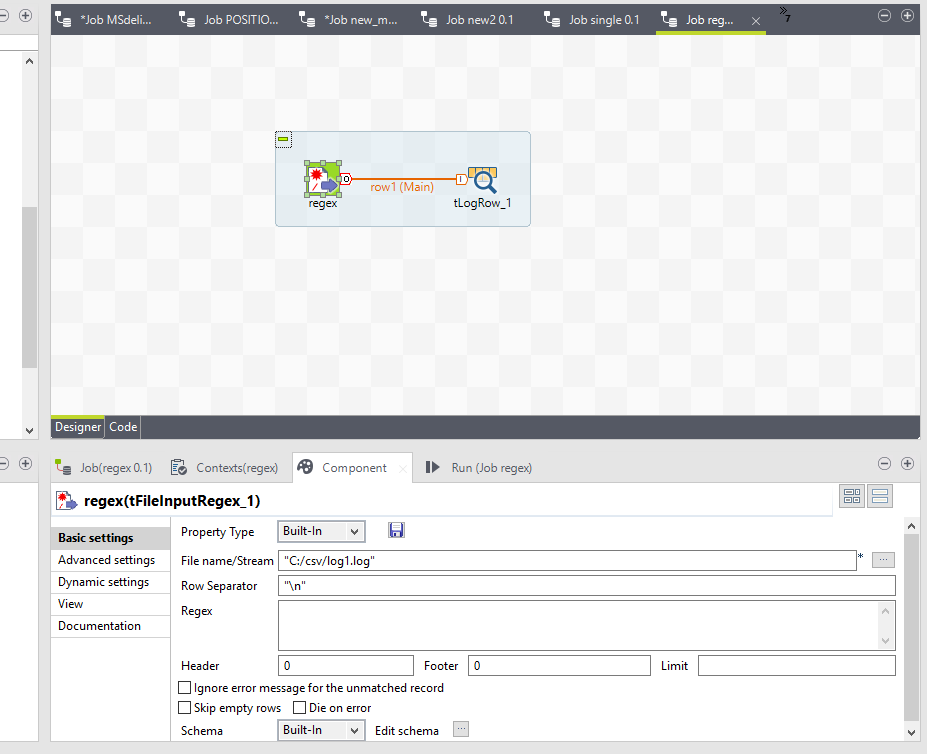
* Regexmetadata file
* Tlogrow

**Procedure:**

* 1. Just create a metadata repository of regex file and just grag it to workspace
  2. Then double click it and browse the log file
  3. And then configure the regex and add it to tlog row

**OUTPUT:**

****

****

1. **How to Write Data to a Positional File?**

**Components Required:**

* **tFileInputMSPositional**
* **tFileOutputDelimited**

**Procedure:**

### Scenario: Reading data from a positional file

The following scenario write data to a positional file, which contains two schemas. The positional file is shown below:

schema\_1 (car\_owner):schema\_id;car\_make;owner;age

schema\_2 (car-insurance):schema\_id;car\_owner;age;car\_insurance

1bmw John 45

1bench Mike 30

2John 45 yes

2Mike 50 No

#### Dropping the components

1. Drop one **tFileInputMSPositional** and two **tLogRow** from the **Palette** to the design workspace.
2. Renamethetwo **tFileOutputDelimited** components as **car\_owner** and **car\_insurance**.

#### Configuring the components

1. Double-click the **tFileInputMSPositional** component to show its **Basic settings** view and define its properties.
2. In the **File name/Stream** field, type in the path to the input file. Also, you can click the **[...]** button to browse and choose the file.
3. In the **Header Field Position** field, enter the start-end position for the schema identifier in the input file, 0-1 in this case as the first character in each row is the schema identifier.
4. Click the **[+]** button twice to added two rows in the **Records** table.
5. Click the cell under the **Schema** column to show the **[...]** button.

Click the **[...]** button to show the schema naming box.

1. Enter the schema name and click **OK**.

The schema name appears in the cell and the schema editor opens.

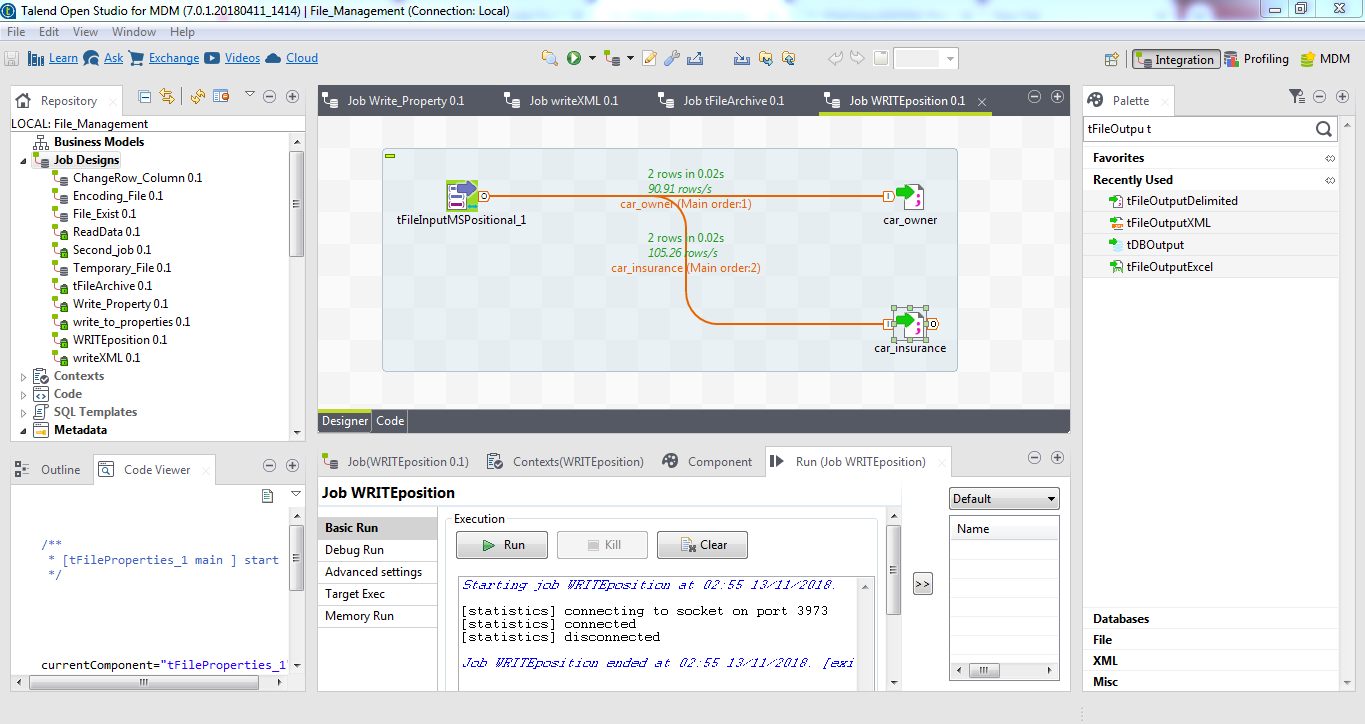
1. Define the schema car\_owner, which has four columns: schema\_id, car\_make, owner and age.
2. Repeat the steps to define the schema car\_insurance, which has four columns: schema\_id, car\_owner, age and car\_insurance.
3. Connect **tFileInputMSPositional** to the **car\_owner** component with the **Row** > **car\_owner** link, and the **car\_insurance** component with the **Row** > **car\_insurance** link.
4. In the **Header value** column, type in the schema identifier value for the schema, 1 for the schema car\_owner and 2 for the schema car\_insurancein this case.
5. In the **Pattern** column, type in the length of each field in the schema, the number of characters, number, etc in each field, 1,8,10,3 for the schema car\_owner and 1,10,3,3 for the schema car\_insurance in this case.
6. In the **Skip from header** field, type in the number of beginning rows to skip, 2 in this case as the two rows in the beginning just describes the two schemas, instead of the values.
7. Choose **Table (print values in cells of a table)** in the **Mode** area of the components **car\_owner** and **car\_insurance**.

#### Executing the Job

1. Press **Ctrl+S** to save the Job.
2. Press **F6** or click **Run** on the **Run**tab to execute the Job.

The file is writing row by row based on the length values defined in the **Pattern** field and output in two tables with different schemas.

**OUTPUT:**

****

1. **How to Write Data to a XML File?**

**Components Required:**

* **tFileInputPositional**
* a**tFileOutputXML**

**Procedure:**

From Positional to XML file

The following scenario describes a two-component Job, which aims at reading data from an input file that contains contract numbers, customer references, and insurance numbers as shown below, and outputting the selected data (according to the data position) into an XML file.

Contract CustomerRef InsuranceNr

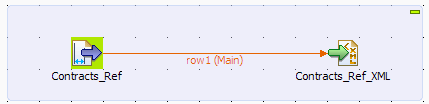
00001 8200 50330

00001 8201 50331

00002 8202 50332

00002 8203 50333

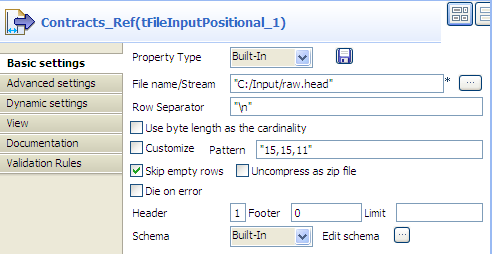
*Dropping and linking components*



1. Drop a **tFileInputPositional** component from the **Palette** to the design workspace.
2. Drop a **tFileOutputXML** component as well. This file is meant to receive the references in a structured way.
3. Right-click the **tFileInputPositional** component and select **Row** *>* **Main**. Then drag it onto the **tFileOutputXML** component and release when the plug symbol shows up.

*Configuring data input*

1. Double-click the **tFileInputPositional** component to show its **Basic** **settings** view and define its properties.



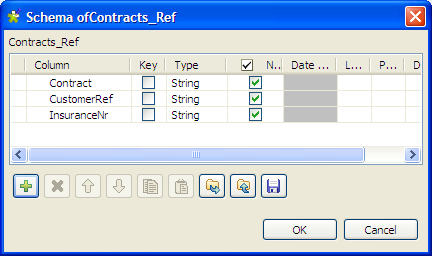
1. Define the Job Property type if needed. For this scenario, we use the built-in Property type.

As opposed to the Repository, this means that the Property type is set for this station only.

1. Fill in a path to the input file in the **File Name** field. This field is mandatory.
2. Define the **Row separator** identifying the end of a row if needed, by default, a carriage return.
3. If required, select the **Use byte length as the cardinality** check box to enable the support of double-byte character.
4. Define the **Pattern** to delimit fields in a row. The pattern is a series of length values corresponding to the values of your input files. The values should be entered between quotes, and separated by a comma. Make sure the values you enter match the schema defined.
5. Fill in the **Header**, **Footer** and **Limit** fields according to your input file structure and your need. In this scenario, we only need to skip the first row when reading the input file. To do this, fill the **Header** field

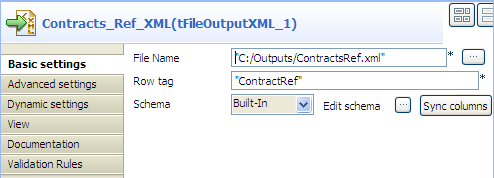
with *1* and leave the other fields as they are.

1. Next to **Schema**, select **Repository** if the input schema is stored in the Repository. In this use case, we use a **Built-In** input schema to define the data to pass on to the **tFileOutputXML** component.
2. You can load and/or edit the schema via the **Edit Schema** function. For this schema, define three columns, respectively *Contract*, *CustomerRef* and *InsuranceNr* matching the structure of the input file. Then, click **OK** to close the **[Schema]** dialog box and propagate the changes.

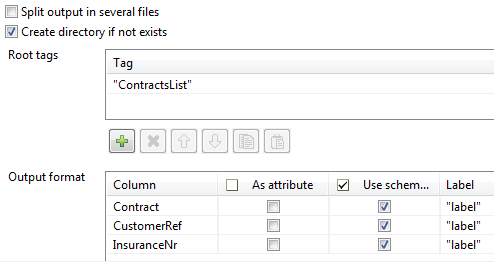


*Configuring data output*

1. Double-click **tFileOutputXML** to show its **Basic settings** view.



1. Enter the XML output file path.
2. Define the row tag that will wrap each row of data, in this use case *ContractRef*.
3. Click the three-dot button next to **Edit schema** to view the data structure, and click **Sync columns** to retrieve the data structure from the input component if needed.
4. Switch to the **Advanced settings** tab view to define other settings for the XML output.

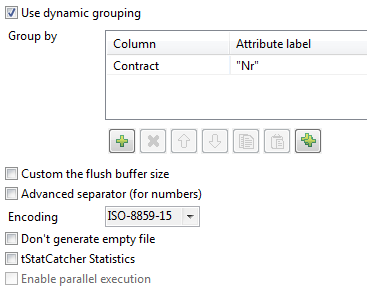


1. Click the plus button to add a line in the **Root tags** table, and enter a root tag (or more) to wrap the XML output structure, in this

case *ContractsList*.

1. Define parameters in the **Output format** table if needed. For example, select the **As attribute** check box for a column if you want to use its name and value as an attribute for the parent XML element, clear the **Use** **schema column name** check box for a column to reuse the column labelfrom the input schema as the tag label. In this use case, we keep all the default output format settings as they are.
2. To group output rows according to the contract number, select the **Use** **dynamic grouping** check box, add a line in the **Group by**table,

select **Contract** from the **Column** list field, and enter an attribute for it in the **Attribute label** field



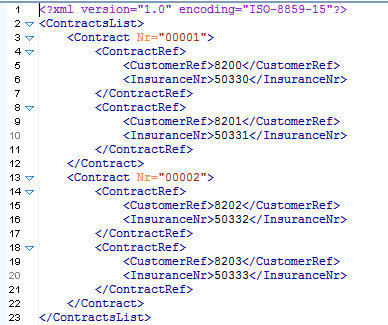
9. Leave all the other parameters as they are.

*Saving and executing the Job*

1. Press **Ctrl+S** to save your Job to ensure that all the configured parameters take effect.
2. Press **F6** or click **Run** on the **Run** tab to execute the Job.

The file is read row by row based on the length values defined in the **Pattern** field and output as an XML file as defined in the output settings. You can open it using any standard XML editor.

**OUTPUT: XML File:**



1. **How to Write Data To a Multi Structured Positional File?**

**Components Required:**

* **tFileInputMSPositional**
* **tFileOutputDelimited**

**Procedure:**

### Scenario: Reading data from a positional file

The following scenario write data to a positional file, which contains two schemas. The positional file is shown below:

schema\_1 (car\_owner):schema\_id;car\_make;owner;age

schema\_2 (car-insurance):schema\_id;car\_owner;age;car\_insurance

1bmw John 45

1bench Mike 30

2John 45 yes

2Mike 50 No

#### Dropping the components

1. Drop one **tFileInputMSPositional** and two **tLogRow** from the **Palette** to the design workspace.
2. Renamethetwo **tFileOutputDelimited** components as **car\_owner** and **car\_insurance**.

#### Configuring the components

1. Double-click the **tFileInputMSPositional** component to show its **Basic settings** view and define its properties.
2. In the **File name/Stream** field, type in the path to the input file. Also, you can click the **[...]** button to browse and choose the file.
3. In the **Header Field Position** field, enter the start-end position for the schema identifier in the input file, 0-1 in this case as the first character in each row is the schema identifier.
4. Click the **[+]** button twice to added two rows in the **Records** table.
5. Click the cell under the **Schema** column to show the **[...]** button.

Click the **[...]** button to show the schema naming box.

1. Enter the schema name and click **OK**.

The schema name appears in the cell and the schema editor opens.

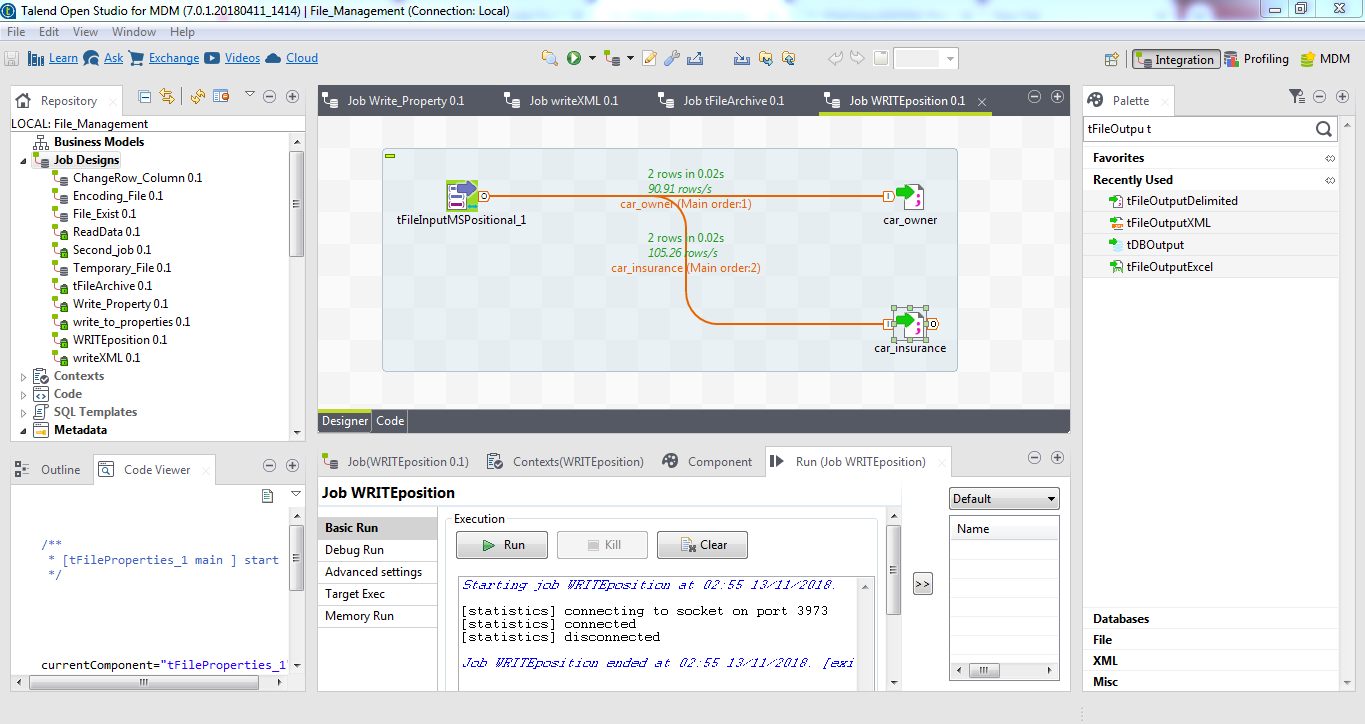
1. Define the schema car\_owner, which has four columns: schema\_id, car\_make, owner and age.
2. Repeat the steps to define the schema car\_insurance, which has four columns: schema\_id, car\_owner, age and car\_insurance.
3. Connect **tFileInputMSPositional** to the **car\_owner** component with the **Row** > **car\_owner** link, and the **car\_insurance** component with the **Row** > **car\_insurance** link.
4. In the **Header value** column, type in the schema identifier value for the schema, 1 for the schema car\_owner and 2 for the schema car\_insurancein this case.
5. In the **Pattern** column, type in the length of each field in the schema, the number of characters, number, etc in each field, 1,8,10,3 for the schema car\_owner and 1,10,3,3 for the schema car\_insurance in this case.
6. In the **Skip from header** field, type in the number of beginning rows to skip, 2 in this case as the two rows in the beginning just describes the two schemas, instead of the values.
7. Choose **Table (print values in cells of a table)** in the **Mode** area of the components **car\_owner** and **car\_insurance**.

#### Executing the Job

1. Press **Ctrl+S** to save the Job.
2. Press **F6** or click **Run** on the **Run**tab to execute the Job.

The file is writing row by row based on the length values defined in the **Pattern** field and output in two tables with different schemas.

**OUTPUT:**

****

1. **How to Write Data To a Multi Structured XML File?**

**Components Required:**

tFileOutputMSXML

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  | **Function** | |  |  | **tFileOutputMSXML** writes multiple schema within an XML |  |  |
|  |  |  |  |  |
|  |  |  |  |  | structured file. |  |  |
|  |  |  |  |  |  |  |  |
|  | | | |  |  | |  |
|  | **Purpose** | |  |  | **tFileOutputMSXML** creates a complex multi-structured XML file, |  |  |
|  |  |  |  |  | using data structures (schemas) coming from several |  |  |
|  |  |  |  |  | incoming **Row** flows. |  |  |
|  |  |  |  |  |  |  |  |